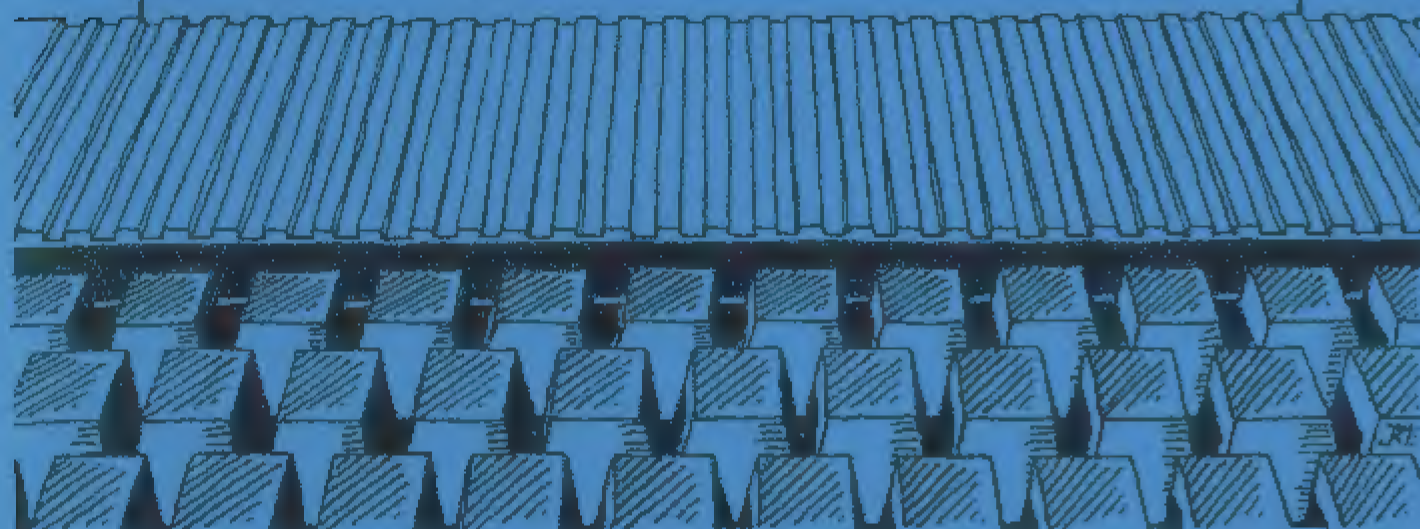
Acorn Nieuws

Jaargang
nummer



Voorz./penningm.
Th. van Kempen.
Het Puyven 71.
5672 RB Nuenen.
Telf 040 - 836210

Secretaris:
A. Jongeling.
Klokkengietershoeve 3
7326 SB Apeldoorn.
Telf 055 - 417314

Ledenadministratie:
G. Visser.
Haringvliet 391.
8032 HL Zwolle.
Telf 038 - 546561.

Uitsluitend voor betaling van contributie:
Giro 5244293 Bank 93.32.87.283
Beide t.n.v. Penningmeester Acorn computerclub.
Eindhoven.

Uitsluitend voor levering van clubartikelen:
BANK 52.84.69.010 t.n.v. Acorn computerclub.
Eindhoven.
Postgiro van de bank: 1150000 ABN Eindhoven.

Redactie ACORN NIEUWS:
Rudi van Drunen - hardware
Frans van Hoesel - assembler
Hans Marks - basic
Henk Reinders - layout

Redactie Acorn Nieuws:
Postbus 1050
9700 BA Groningen.
Telf 050 - 125458

Bandjesarchief:
P. Greveit.
Swalmstraat 12
1784 CP Den Helder
Telf 02230 - 37060

Datasheets:
G. Akkermans.
Wikke 1
1273 BR Huizen.
Telf 02152 - 60294.

Drukwerkarchief:
F. Monsato
Biesbosch 153
8032 VD Zwolle.

Uiterste datum inlevering copy:

NR 6. 24 - 8 - 1984.
NR 7. 26 -10 - 1984.
NR 1. 10 -12 - 1984.

DE CLUB-WINKEL:

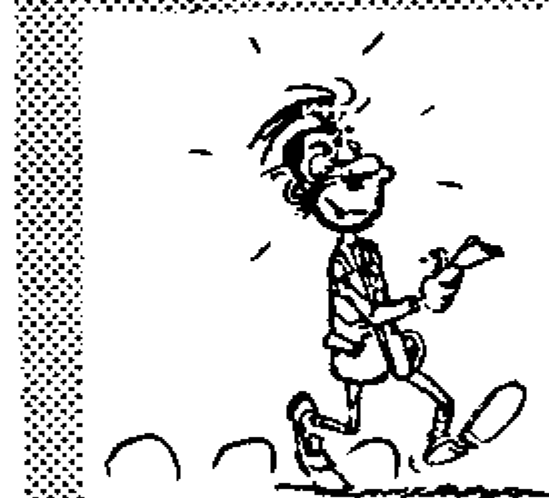
Geheugenkaart; 16 kByte extra in de ATOM	fl. 50,00
Schakelkaart; meerdere EPROM's op Axxx (incl.74LS133)	fl. 50,00
Programmerkaart; zelf programmeren van EPROMs	fl. XX,XX
Big Benny print;incl. Clock IC MSM 5832RS	fl. 42.50
Kleurenkaart uit DELFT (nog beperkt aanwezig)	fl. 17.00
Herdruk ACORN NIEUWS 1982; 97 pg. wetenswaardigheden	fl. 6.00
Jaargang 1983: totaal ruim 450 pag.	fl. 30.00
ATOM-WARE deel 1; machinetaal op de ATOM 98 pag.	fl. 6.00

Bestellen:
Bij Uw regionale penning-meester.
Rechtstreeks bij de federatieve penningmeester, dan fl. 4.00 extra voor porto-kosten.
Archiefdiensten:
Voor bandjes, datasheets, EPROMS en alle andere vormen van dienst verlening kunt U terecht bij de regionale archiefdiensten. Mocht in Uw regio iets ontbreken, dan helpt het regionale bestuur U verder op weg.

ACORN NIEUWS Uitgave van de Fed. Acorn Computerclubs N/B
Verschijnt 6 - 8 keer per jaar.

3	SEPT	1984	ARNHEM	t.o. POSTGIRO VELPERWEG 56-A (kelder)
20.00 UUR				ARNHEM.
3	SEPT	1984	DELFT	E-KAFE GEBOUW ELEKTRA TH WIJK
20.00 UUR				DELFT.
6	SEPT	1984	BRABANT	GEBOUW B.R.A.N. HEVEL 7
20.30 UUR				VORSTENBOSCH.
7	SEPT	1984	LIMBURG	ZAAL HUIVENEERS STATIONSSTRAAT 35
19.30 UUR				SITTARD.
12	SEPT	1984	TWENTE	MEPA-GEBOUW WIERDENSESTRAAT 40
20.00 UUR				ALMELO.
13	SEPT	1984	NOORD	MUNSTERHOES. MUNSTERHEERD.
19.30 UUR				GRONINGEN.
18	SEPT	1984	BRABANT	DE WERFF V/D WERFFSTRAAT
20.00 UUR				EINDHOVEN.
19	SEPT	1984	CENTRUM	DE LANTAERN UTRECHTSESTRAATWEG 4
20.00 UUR				NIEUWEGEIN.
22	SEPT	1984	BELGIE	OXACO-CEPENBERGCENTER BORSBEEKSE STEENWEG 45.
10.30 UUR				BOECHOUT (BIJ HOVE)
24	SEPT	1984	ROTTERD	SPEELTUINGEBOUW HET WESTEN SPAANSEWEG 38
20.00 UUR				ROTTERDAM-WEST.
29	SEPT	1984	OV/GELD	WIJKCENTRUM DE BOLDER. DOBBE 62.
14.00 UUR				ZWOLLE.
1	OKT	1984	ARNHEM	t.o. POSTGIRO VELPERWEG 56-A (kelder)
20.00 UUR				ARNHEM.
1	OKT	1984	DELFT	E-KAFE GEBOUW ELEKTRO TH WIJK
20.00 UUR				DELFT.
4	OKT	1984	BRABANT	GEBOUW B.R.A.N. HEVEL 7
20.30 UUR				VORSTENBOSCH.

REDACTIONEEL



Te veel. Op deze manier blijft er zelfs geen ruimte over voor een behoorlijk redactioneel. Niet dat dat belangrijk is, want wie interesseert zich nu voor de redactionele perikelen van dit blad. Geen hond toch (woef).
Maar goed, in ieder geval hebben jullie weer wat te lezen. Misschien zijn jullie in de vakantie ook nog toe gekomen aan het schrijven van iets leuks. Wat zei je? Slecht weer geweest? Zo, zo, dus tijd genoeg gehad om achter het toetsenbord te zitten. Waarom dan geaarzeld met het insturen van dat interessante, originele, moeilijke, gekleurde, korte, uiterst simpele, of lachwekkende programma of artikeltje voor Acorn Nieuws? Wij wachten!!!
Overigens krijgen wij soms klachten over het niveau van bepaalde artikelen. vaak in de trant van: "Ik begrijp het wel, maar ik ken mensen die...". Kreten als "te moeilijk" of "te weinig uitleg" horen wij dan het vaakst. Beste mensen bedenk dat zeker 80% van de pagina's in dit blad van een dusdanig niveau zijn dat iedereen die min of meer moet kunnen volgen en dat u waarschijnlijk over een tijdje met plezier die andere 19% leest. Dat er in sommige gevallen te weinig wordt uitgelegd realiseren wij ons ook, doch het is voor de redactie ondoenlijk om regelmatig artikelen aan te vullen met opmerkingen. Wij zijn geheel afhankelijk van wat de leden ons insturen. Bovendien, als we van ons ieder persoonlijk teveel plaatsen krijgen we daar weer klachten over. Dit laatste is trouwens een verschijnsel dat ook in de regio-bladen is te bespeuren: redakteuren schrijven meer.
Over het algemeen zijn de reacties op Acorn Nieuws echter goed. Dit is echter niet zozeer een verdienste van de redaktie, alswel van de vele leden die zich actief met de atom bezig houden en daarover ook iets zinnigs weten te vertellen. Graag zouden wij meer van dit soort leden hebben. Bij deze is de redactie dus op een topsterkte gekomen van 800 man.
Wij wachten, er zijn er Veel te Weinig.

de redaktie.

BESTUUR:

Theo van Kempen heeft de financiele administratie overgenomen.
Gerhard Visser blijft de ledenadministratie doen.
Adreswijzigingen en niet ontvangen A.N. dient aan Gerhard Visser te worden gemeld en NIET aan de redaktie.

VERSCHIJNING A.N.

Helaas is het financieel niet haalbaar A.N. acht maal uit te geven met 100 pagina's. Daarom zal het november nummer vervallen en het oktobernummer omstreeks 15 oktober verschijnen.

CONTRIBUTIE 1985:

Een acceptgirokaart voor de contributie zal bij het oktobernummer gevoegd worden. Op de Algemene Vergadering is deze vastgesteld op fl. 60,00. Wij hopen dan volgend jaar wel 8 nummers uit te geven. De contributie dient voor 15 december 1984 voldaan te zijn om het januari nummer te ontvangen. Bij betaling na deze datum zal het januari-nummer niet gratis nagestuurd worden, daar deze kosten de pan uit rijzen, maar dient er fl. 4,00 extra betaald te worden.

PRINTEN:

Het Big Benny printje is leverbaar inclusief het clock-IC vanaf heden.
De opdracht voor het maken van schakel- en geheugenkaarten is gegeven, leverbaar vanaf eind september.
Ook zal er een offerte gevraagd worden voor het minischakelkaartje.
Uit de enquete is gebleken dat er veel vraag naar is.
Van de kleuren kaart uit DELFT zijn nog 5 exemplaren leverbaar. De verzending van de clubartikelen wordt verzorgd door Theo van Kempen.

ATOMWARE:

De voorbereiding van deel 2 vergt meer tijd dan was voorzien en de hoeveelheid is dermate groot dat de zaak in deel 2 (dosrom) en deel 3 (monitorrom) gesplitst moet worden. De uitgave is daarom nog niet exact aan te geven, maar we doen ons best.
Evert Sanders uit Roermond is vol ijver de manual aan het vertalen en toe te lichten. Dit zal ook worden uitgegeven als een ATOM-WARE deeltje.

REGID-SCHIJVEN:

Inhoud 1 september 1984:

NEDnord	2200	0000	06000	002	superdraw tekening.
NEDzuid	2200	0000	06000	062	superdraw tekening.
CROSS	2200	0000	06000	0C2	superdraw tekening.
RAADSEL	2900	AFAF	001E8	122	A.N. nr 5 blz. 53.
SOS.CK	2900	AFAF	03259	124	A.N. nr 5 blz. 59.

Inhoud 2 september 1984:

ZEESLAG	2900	AFAF	0114A	002	A.N. nr 5 blz. 28.
GRMOD	2900	AFAF	01652	014	A.N. nr 5 blz. 32.
PLAY	2900	AFAF	01652	02B	A.N. nr 5 blz. 34.
FIND	2900	AFAF	00365	042	A.N. nr 5 blz. 35.
SCREENS	2900	AFAF	00314	04E	A.N. nr 5 blz. 38.
F-CAT1	2900	AFAF	00175	04A	A.N. nr 5 blz. 49.
SETCAT	2900	AFAF	00175	04C	A.N. nr 5 blz. 49.

EXEC$	2900	AFAF	00293	04E	A.N. nr 5 blz.
DOBBEL	2900	C2B2	00E87	051	A.N. nr 5 blz. 75.
TURTLE	2900	C2B2	00E76	060	A.N. nr 5 blz. 85.
HANOI	2900	C2B2	0038A	067	A.N. nr 5 blz. 84.
DRIEHOE	2900	C2B2	00461	06B	A.N. nr 5 blz. 88.
H-BREEK	2900	C2B2	004B6	070	A.N. nr 5 blz. 86.
3-D-R	2900	C2B2	005BD	075	A.N. nr 5 blz. 87.
KOPPEL	2900	C2B2	002E5	07B	A.N. nr 5 blz. 48.
65C02	2900	AFAF	002E5	07E	A.N. nr 5 blz. 17.
APPEND	2900	AFAF	00414	081	A.N. nr 5 blz. 37.
CHECK	2900	AFAF	0032A	086	A.N. nr 5 blz. 40.
GFILL	2900	AFAF	0073E	08A	A.N. nr 5 blz. 21.
GF-MOD4	2900	AFAF	00473	092	A.N. nr 5 blz. 21.
DSIZE	2900	AFAF	0027D	097	A.N. nr 5 blz. 39.
ZUIGMUI	2900	AFAF	01063	09A	A.N. nr 5 blz. 41.
ATCALC	8200	8202	01000	0AB	A.N. nr 5 blz. 10.
MCAT	2900	AFAF	00427	0BB	A.N. nr 5 blz. 52.
MDCR-2.	2900	AFAF	03E99	0C0	A.N. nr 5 blz. 61.
SEARCH	2900	AFAF	0034D	0FF	A.N. nr 5 blz. 36.

DE VOORPLAAT:
De voorplaat van nummer 4 was gemaakt door ANDRE DE BRUIN hetgeen verzuimd is te melden, waarvoor excuses. De voorplaat van dit nummer komt uit Limburg en is gemaakt door HANS HEUTS.

DE ENQUETE:
De uitwerking van de enquete eist meer tijd dan was gepland, zodat U de uitslag in het volgende nummer kunt aantreffen. Ongeveer 300 formulieren mochten wij retour ontvangen, zodat 30% gereageerd heeft. Dit is een bijzonder hoge score waarvoor onze dank. De opmerkingen, die op een uitzondering na, positief waren, zullen wij ter harte nemen.

REGIOBLADEN:

DE CURSOR nr 4. (brabant)
Dobbelspel; C-K Operating system; Procedures en functie's koppelen; Diverse programma's; Commando's voor de inktjet printer.

BRONSGROEN EIKELTJE nr. 4 (limburg)
Vertaling manual van het COS gebeuren; LF-frequentiemeter; Talstelsels; gewijzigde reset-routine; Statement PLIST; Wordpack wijziging; Printercodering voor de Gemini 10X; 3 Morse programma's.

BRONSGROEN EIKELTJE nr. 5 (limburg)
Een oplossing om de ruis van het scherm te laten verdwijnen; Vertaling manual arrays en vectoren; Telroutine voor infomaster; ATOM STEREO geluids processor

ERROR 34 nr. 3 (noordholland)
GFILL-statement; Computerjargon verklaard; Programma's CHECK, APPEND en VERIFY; DISKDRIVE problemen
DATACHECK nr. 4 (den haag)
Vervolg van het ASBK schakelsysteem; Assembler voor de 65C02; enige wist U datjes

In de ontvangen post werden enige vragen gesteld, waarvan de antwoorden voor anderen ook belangrijk kunnen zijn.

1. In het sector programma kan de subroutine #E223 verwijderd worden. Deze subroutine zet de catalog in het geheugen. Wanneer in de eerste twee sectoren op schijf zich een DISK-ERROR bevindt, wordt het programma voortijdig beeindigd aangezien hierin zich de catalog in bevind.

2. Bij het gebruik van toolboxen kan het gebeuren dat de file "vreemd" op schijf terecht komt en niet meer te laden is. Dit vreemde is echter niet zo vreemd, maar is het gevolg dat vele toolboxen met DOS zeropage adressen werken. Vooral wanneer #AC wordt gebruikt, waarin de qualifier staat waaronder de file op schijf wordt gezet. Door deze qualifier met *SET op de waarde te zetten van de file op schijf, kan de file weer gelezen worden.

3. Diverse problemen met de orginele ATOM DISKDRIVE waarin een OLIVETTI drive wordt gebruikt, ligt meesal aan de drive. Een pasklaar antwoord is moeilijk aan te geven. Wel is bekend dat er een aantal zijn geleverd met een voeding voor aansluiting op 240 VOLT hetgeen in Engeland gebruikt wordt. Zodoende wordt er geen 5 en 12 volt geproduceerd. Ook schijnt er een microswith voor track 0 detectie niet feilloos te werken. Temperatuurgevoeligheden zijn ook reeds gemeld. Het probleem wordt dan alleen maar groter door het afgeven van DISK-ERRORS welke niet aangeven wat er met de drive fout is, maar alleen aangeven wat de DOSCONTROLlER heeft geconstateerd.

4. Over de variabelen kwam een vraag wat X=%X/1 betekent. X is een integere variabele waarvan de berekening in de basic-interpreter geschiedt. %X is een floating point variabele waarvan de berekening via de floating point gaat. Beide variabelen hebben een eigen stack. Met X=%X wordt de waarde van de floating point variabele aan de integere variabele toegekend. De integere variabele wordt hierbij ontdaan van de waarden achter de komma.

5. Met PUT, BPUT en SPUT wordt iets in een random file gezet. Met GET, BGET en SGET wordt iets uit een random file gelezen. Indien de volgorde bij het lezen van een random file, anders is dan bij het schrijven, krijgt men verkeerde waarden. Dus bijv. bij het schrijven 5 x PUT, 4 x BPUT en 3 x PUT moet gelezen worden met 5 x GET, 4 x BGET en 3 x GET. Gaat men nu lezen met 3 x GET, 4 x BGET en 5 X GET, dan krijgt met andere waarden dan men heeft geschreven. Dit is het gevolg van een foutieve programmering.

6. Vele problemen waarvoor gebeld wordt zijn terug te voeren tot "TOP" problemen. Bij het inladen met DOS of COS statements wordt de TOP van het programma NIET goed gezet. Dit doet men met het statement OLD of END. De basic-statements voor DOS of COS gebruik zetten de TOP wel goed. MAAR deze statements mogen niet in een programma gebruikt worden omdat zijn als startadres het begin van het programma gebruiken en als eindadres detop van het programma.

7. CHARON is een programma dat de schrijf en leesvectoren constant wijzigd. Dit kan problemen veroorzaken met programma's die dit ook doen. CHARON dient dan te worden uitgezet met !#208=#FE94FE52.

8. De catalog van een disk kan worden geledigd met:
?#2105=00;LINK #E75B;LINK #E74A.

Indien U ook vragen heeft stuur ze dan en wij zullen ze in ACORN NIEUWS beantwoorden.

```
  2REMvoor scr.d. maak regel 0 39;verwijder 0 en jump nr 39
  3REMvoor fully scr.d. in 39: U=400 EN V=3 EN W=2
  4REMregel115 is einde
  5DATA48,62,0,84,22,58,0,60,22,42,0,30
 10DATA22,22,0,5,42,22,15,0,62,22,30,0,82,22,60,0,102,22,100,0
 15DATA122,22,120,0,142,22,145,0,162,22,170,0,180,22,190,0
 20DATA200,22,220,0,204,42,250,0,204,62,255,30,204,82,255,60
 25DATA204,102,255,90,204,122,255,120,204,142,255,155
 30DATA190,146,255,185,170,146,230,191,150,146,140,191
 35DATA150,126,160,191,150,106,190,191,150,86,215,191;REMr0lyn
 39U=400;V=3;W=2;REM////I.C.PLOTTING///HAHE.'83////
 40aCLEAR4;GOS.i;K=5
 45F.Q=0TO U;B=A.R.%210+20;C=A.R.%150+20;D=A.R.%210+20
 50PLOT4,B,C;PLOT6,D,C;PLOT6,D,(B/2);N.;U=U+50;IFU=450U=100
 55E=0;DOPLOT4,(10+E),(10+E);PLOTK,(230-E),(10+E);REMkader
 60PLOTK,(230-E),(170-E);PLOTK,(10+E),(170-E)
 65PLOTK,(10+E),(10+E);E=E+1;IFE=9K=7
 70U.E=11;K=5;REMnetwerk
 75F.Q=20TO140;PLOT4,Q,80;PLOT7,Q,160;N.
 80F.Q=20TO140S.V;PLOT4,Q,80;PLOT6,Q,160;N.
 85F.Q=80TO160S.W;PLOT4,20,Q;PLOT6,140,Q;N.;PLOT4,20,80
 90PLOT5,140,80;PLOT5,140,160;PLOT5,20,160;PLOT5,20,80
 95V=V-1;W=W-1;IFW<2V=9;W=8
100RES.;F.O=0TO23;READX,Y,P,Q
105F.Z=0TO13;PLOT4,(X+Z),Y;PLOT5,(X+Z),(Y+13);N.;REMvierkant
110PLOT4,X,Y;PLOT3,13,0;PLOT3,0,13;PLOT3,-13,0;PLOT3,0,-13
115PLOT4,(X+6),(Y+6);PLOT6,P,Q;N.;F.Q=0TO299;WAIT;N.;G.a;HAHE.
120iF.Q=183TO187;PLOT4,20,Q;PLOT5,60,Q;N.;F.Q=183TO187
125PLOT4,33,Q;PLOT7,47,Q;N.;PLOT4,18,181;PLOT5,62,181;REMcha
130PLOT5,62,189;PLOT5,18,189;PLOT5,18,181;PLOT4,35,183
135PLOT5,37,183;PLOT4,36,183;PLOT5,36,187;PLOT4,35,187
140PLOT5,37,187;PLOT4,44,183;PLOT5,41,183;PLOT5,41,187
145PLOT5,44,187;PLOT13,39,183;PLOT13,46,183;R.
```

Een vergissing is menselijk,maar om er echt een puinhoop van te maken, heb je een komputer nodig

ATOM-CALC

ATOM-CALC IS DE ACORNSOFT VERSIE VAN EEN SPREADSHEET OF EEN VISICALC.
DIT PROGRAMMA, WORDT GELEVERD IN EEN 2532 EPROM (VOOR OP DE PLAATS VAN IC-24), KAN JE BESCHOUWEN ALS EEN CROOT (EN DUUR) KLADBLOK. DIT KLADBLOK IS INGEDEELD IN VAKJES:

```
  62 HORIZONTAAL  A-BK
 255 VERTIKAAL    1-255
```

HET VAKJE LINKSBOVEN IS DAN A-1 EN HET VAKJE RECHTSONDER IS DAN DUS BK-255.

HET BEELDSCHERM IS EEN VENSTER(TJE) OP DIT KLADBLOK WAARMEE WE EEN STUKJE VAN HET KLADBLOK KUNNEN BEKIJKEN. DE BOVENSTE 3 REGELS VAN HET SCHERM ZIJN GERESEVEERD VOOR DE HEADER.
DE PROMPT IS EEN NAAR LINKS WIJZEND PIJLTJE, EN DE CURSOR IS EEN BALK VAN 8 KARAKTERS LANG DIE ZICH GEDRAAGT ALS DE NORMALE ATOM CURSOR. HET BEELDSCHERM KAN IN ALLE RICHTINGEN SCROLLEN MITS JE NIET BIJ DE RAND VAN HET KLADBLOK ZIT WANT JE KAN NIET V WANT JE KAN NIET (LOGIES).
IN DE LINKER BOVEN HOEK VAN HET SCHERM (IN DE HEADER DUS) ZIE JE DE COORDINATEN VAN HET VAKJE WAAR DE CURSOR ZICH BEVINDT. DAAR ONDER ZIE JE DE INHOUD VAN DAT VAKJE (NIET DE NUMERIEKE WAARDE VAN HET VAKJE, DEZE ZIE JE OP HET SCHERM WAAR DE CURSOR ZICH BEVINDT). DAARONDER (DUS OP DE DERDE REGEL) STAAT DE CURSOR EN DAAR VERSCHIJNT ALLES WAT JE INTIKT (MAX.32 TEKENS) ALS MEN OP RETURN DRUKT DAN WORDT WAT JE NET HEBT INGETIKT OP DE TWEEDE REGEL GEZET EN ALS HET MOGELIJK IS DAN ZET ATOM-CALC DE NUMERIEKE WAARDE HIERVAN O(DE PLAATS WAAR DE CURSOR STAAT. DAT VAKJE BEVAT NU DUS WAT JE NET INTIKTE.

OH JA, HET OPSTARTEN.....
ALS JE IN (ATOM) DIRECT MODE 'CALC' INTIKT DAN WORDT ATOM-CALC OPGESTART. HET GEHEUGEN WORDT NU SCHOONGEMAAKT EN DE HEADER VERSCHIJNT MET EEN COPYRIGHT MELDING. OM ATOM-CALC TE VERLATEN MOET JE OP BREAK DRUKKEN.
MAAR OH JEE, JE DRUKT PER ONGELUK OP BREAK EN JE HEBT DIE HELE MIDDAG JE KASBOEK IN ZITTEN TE TIKEN! ALS JE NU 'CALC' INTIKT DAN WORDT JE WERK TENIET GEDAAN. NOU GEEN PANIEK, TIK DAN 'CALCR' IN EN DRUK DAN OP LOCK (DUS ALS JE AL WEER IN ATOM-CALC ZIT) EN ALLES IS WEER ZOALS HET WAS VOORDAT JE OP BREAK DRUKTE (MOOI HE).

HET INVOEREN VAN DATA:
ATOM-CALC KAN 3 SOORTEN VAN INVOER AAN NL.:

```
  LABELS       BV.   FRED
  GETALLEN     BV.   2001
  FORMULES     BV.   (A-1+3*B3)
           OF  BV.   (SIN(A3))
```

EEN VOORBEELD:

START MET EEN SCHONE LEI (DUS BREAK EN 'CALC').
DE CURSOR STAAT NU OP A-1.
TIK NU FRED IN. IN VAKJE A-1 KOMT NU FRED TE STAAN. OP REGEL 1 IN DE HEADER VERSCHIJNT NU EEN 'L' (VOOR LABEL). HAD JE NU EEN GETAL INGETIKT INPLAATS VAN FRED DAN HAD ER OP REGEL 1 IN DE HEADER EEN 'V' GESTAAN (V VOOR VALUE, EEN WAARDE DUS OFTEWEL EEN GETAL OF EEN FORMULE)
EEN ALFANUMERIEKE STRING (DUS EEN STRING DIE LETTERS EN/OF GETALLEN BEVAT) BEHALVE DAN COORDINATEN VAN DE VAKJES EN DE FORMULES, BEHANDELT ATOM-CALC ALS EEN LABEL.
ALS EEN INPUT STRING NIET ALS EEN VALUE HERKEND WORDT DAN IS HET EEN LABEL EN KAN ATOM-CALC ER NIET MEE REKENEN.

MAAR GOED WE HADDEN FRED OP A-1 GEZET. ZET DE CURSOR NU OP B-1 EN TYPE NU JAN IN. OP B-1 STAAT NU JAN. ZET DE CURSOR NU OP A-2 EN TYPE 10 IN. OP A-2 STAAT NU DE WAARDE 10. ZET DE CURSOR OP B-2 EN TYPE (A2+5) IN. OP B-2 VERSCHIJNT NU 15 EN OP REGEL 3 VAN DE HEADER STAAT NU (A2+5). WE KUNNEN DIT VOORBEELD BESCHOUWEN ALS EEN TABEL VAN HOEVEEL ZAKGELD FRED EN JAN KRIJGEN.
ZET DE CURSOR NU OP C-1 EN TYPE ERIK IN. OP C-2 TYPEN WE NU (A2-5) EN OP C-2 VERSCHIJNT NU DUS 5. NU HEBBEN WE HET ZAKGELD VAN ERIK ER OOK BIJ.
ZET DE CURSOR NU WEER OP A-2 EN TYPE 11 IN. B-2 VERANDERT NU IN 16 EN C 2 VERANDERT NU IN 6. ATOM-CALC REKENT HET ALLEMAAL NETJES DOOR VOOR ONS.
DIT WAS MAAR EEN SIMPEL VOORBEELD. INPLAATS VAN (A2-5) OP C 2 HADDEN WE OOK BV. (SIN(A2*B2/(2*PI))) NEER KUNNEN ZETTEN. WE HADDEN OOK BV. OP K-99 (A2*3) NEER KUNNEN ZETTEN. ALS WE A-2 NU VERANDEREN DAN VERANDERT K-99 OOK NETJES MEE.
ALLE FLOATING POINT WISKUNDIGE FORMULES ZIJN TE GEBRUIKEN NL.:

ABS	ACS	ASN	ATN	COS
DEG	EXP	FLT	LOG	PI
RAD	RND	SGN	SIN	SQR

WAT DEZE FUNTIES ALLEMAAL DOEN STAAT IN ATOMIC THEORY AND PRACTICE.
UITERAARD ZIJN OOK DE GEWONE WISKUNDIGE BEWERKINGEN MOGELIJK, NL.:

+	OPTELLEN	-	AFTREKKEN
*	VERM.	/	DELEN
↑	MACHTSVERHEFFING		

ALLE BEREKENINGEN WORDEN UITGEVOERD MET EEN NAUWKEURIGHEID VAN 31 BITS, TUSSEN 1E-38 EN 1E+38

ALLE ATOM-CALC COMMANDO'S BEGINNEN MET '/'.

/F IS HET FORMAT COMMANDO. HIERMEE GEEF JE AAN HOEVEEL DECIMALEN ER MOETEN WORDEN GEPRINT. NA EEN /F1 WORDT 9.9 AFGEBEELD ALS 9.9. NA EEN /F0 WORDT 9.9 AFGEBEELD ALS 9 EN NIET ALS 10. ATOM-CALC RONDT DUS NIET AF HIJ DRUKT ALLEEN HET OPGEGEVEN AANTAL DECIMALEN AF. VOOR BEREKENINGEN GEBRUIKT ATOM-CALC WEL DE GOEDE WAARDE ONGEACHT DE FORMAT.

/G IS HET GOTO COMMANDO. MET DIT COMMANDO KAN JE SNEL NAAR EEN COORDINAAT SPRINGEN IN PLAATS VAN MET DE CURSOR DAAR NAAR TOE TE

GAAN.

STEL JE HEBT ERGENS STAAN SQR(A-1) EN A-1 IS NEGATIEF DAN ZAL SQR(A-1) ALS EEN LABEL BEHANDELD WORDEN. ALS JE TOCH WILT DAT DIT ALS EEN VALUE BEHANDELD WORDT DAN MOET JE ER EEN '+' VOOR ZETTEN, DUS SQR(A-1) WORDT DAN +SQR(A-1). ALS A-1 DAN NOG NEGATIEF IS DAN ZAL INPLAATS VAN DE WAARDE VAN SQR(A-1) HET WOORD 'ERROR' VERSCHIJNEN OP DE PLAATS VAN DAT VAKJE WAARIN SQR(A-1) STAAT TOTDAT A-1 WEER POSITIEF WORDT.

EEN PAAR VOORBEELDEN VAN TOEGESTANE FORMULES:

```
 1.5000             (1+2)*3/4
 A-1 *6              A-1 + B1 - C1
 (B3 + C1) / D1↑2   (B2 * A4) / 3.3
 A2 * 100 + BJ90    SIN(A-1) * 3
 SQR(ABS(A-1))
```

EEN VOORBEELD VAN FLT EN RND :

```
 FLT (ABS RND % 10)
```

ER VERSCHIJNT NU EEN WILLEKEURIG GETAL TUSSEN 0 EN 10 OP DE PLAATS WAAR DE CURSOR STAAT. ALS JE OP LOCK DRUKT DAN VERSCHIJNT HET VOLGENDE WILLEKEURIGE NUMMER.

COPIEREN VAN BLOKKEN IS OOK MOGELIJK: BV. ZET DE LABEL 'ATOMCALC' OP A-1. TIK NU '/R A1 A1 A1 E255' IN. IN DE HEADER ZAL VOOR ONGEVEER 1 MINUUT HET WOORD COPYING TE ZIEN ZIJN WAARNA 'ATOMCALC' IN ALLE VAKJES ZAL STAAN TUSSEN A-1 EN E-255.
MET /R COPIEER JE DUS 1 BLOK NAAR EEN ANDER BLOK. EEN BLOK WORDT WEERGEGEVEN ALS DE BOVENLINKER COORDINAAT EN DE RECHTERONDER COORDINAAT VAN HET BLOK.
ALS JE /R INTIKT ZONDER COORDINATEN VAN DE 2 BLOKKEN DAN HOOR JE EEN PIEPJE EN KAN JE ALSNOG DE COORDINATEN INTIKKEN.
ALS JE GEWOON / INTIKT DAN VERSCHIJNEN DE COORDINATEN VAN HET VAKJE WAAR DE CURSOR ZICH BEVINDT.
HET IS OOK MOGELIJK OM RELATIEF TE COPIEREN. ALS JE GEWOON INTIKT COPIEER DIT BLOK NAAR DAT BLOK DAN IS DAT ABSOLUUT COPIEREN.
EEN VOORBEELD:
JE WILT A1+1 NAAR 4 ANDERE VAKJES COPIEREN. ZET 1 IN A-1 EN A1+1 IN A2. TYPE NU IN '/R A2 A2 A3 A6'. DAN VOLGT DE VRAAG 'A OR R?' (ABSOLUUT OF RELATIEF).
ALS JE NU A INTIKT DAN GEBEURT ER DIT :

```
 A-1 A-2 A-3 A-4 A-5 A-6
  1   2   2   2   2   2
```

MAAR ALS JE NU R INTIKT DAN GEBEURT ER DIT :

```
 A-1 A-2 A-3 A-4 A-5 A-6
  1   2   3   4   5   6
```

ZIE JE HET VERSCHIL?

WE COPIEERDEN ZOJUIST A-2. A2 BEVATTE A1+1. STEL DAT A-2 (B2+C2) BEVATTE DAN ZOU ATOM-CALC 2 KEER VRAGEN A OR R?. EEN KEER VOOR B-2 EN EEN KEER VOOR C-2.

ALS JE VEEL DATA MOET INVOEREN DAN KAN HET LASTIG ZIJN ALS

ATOM-CALC STEEDS WEER ALLES UITREKENT ALS JE IETS NIEWS INTIKT. OM DIT TE VOORKOMEN, TIK DAN /M (MANUAL) IN. NU REKENT ATOM-CALC NIKS UIT TOTDAT JE OP LOCK DRUKT, WANT DAN REKENT HIJ ALLES WEER UIT.

EEN VOORBEELD:

HIER VOLGT EEN TABEL VAN 12 MAANDEN MET HOEVEEL EENHEDEN ELEKTRICITEIT ER VERBRUIKT IS :

MAAND	EENHEID	EENHEIDSPRIJS	PRIJS
JAN	221.00	0.05	11.05
FEB	187.00		9.85
MAART	257.00		12.85
APRIL	236.00		11.80
MEI	248.00		12.40
JUNI	251.00		12.55
JULI	264.00		13.20
AUG	269.00		13.45
SEP	236.00		11.80
OCT	274.00		13.70
NOV	256.00		12.80
DEC	237.00		11.85

DEZE TABEL WIL JE NU IN ATOM-CALC ZETTEN EN LATEN WE AANNEMEN DAT JE DE LAATSTE KOLOM (PRIJS) NOG NIET HEBT BEREKEND.
IN A-2 TOT A-13 KOMEN JAN TOT DEC, LABELS VOOR DE MAANDEN DUS. IN B-2 TOT B-13 KOMEN DE VERBRUIKTE EENHEDEN VOOR DE BETREFFENDE MAANDEN. OP C-2 KOMT DE KOSTEN PER EENHEID 0.05 (GOEDKOOP HE). OP D-2 KOMT (B2 * C2). EN NU COPIEREN:

```
/R D2 D2 D3 D13
(B2 * C2)  NU R INTIKKEN WANT DIT
  ?        VARIEERT VAN MAAND TOT MAAND
(B2 * C2)  NU A INTIKKEN WANT DIT
      ?    IS 0.05
```

/S HIERMEE SAVE JE ATOM-CALC SHEET OP DISK OF CASSETTE. OP HET BEELDSCHERM VERSCHIJNT :"FILE?". NU TIK JE DE FILENAME IN GEVOLGD DOOR RETURN. NU VERSCHIJNT ER "RECORD TAPE". DRUK OP DE SPATIE BALK, EN ATOMCALC GAAT SAVEN. ER STAAN NU OOK 2 STERRETJES (**) ACHTER DE RECORD TAPE BOODSCHAP. ALS ATOMCALC KLAAR IS MET SAVE DAN VERDWIJNEN DEZE STERRETJES WEER.

/L NET ZOALS /S BEHALVE NU LADEN WE INPLAATS VAN SAVEN.

/P HIERMEE KAN JE EEN SHEET UITLATEN PRINTEN IN TABEL VORM (ZOALS JE HET OP HET BEELDSCHERM ZIET DUS).

/O HIERMEE OUTPUT JE EEN SHEET NAAR DE PRINTER. HET VERSCHIL TUSSEN /O EN /P IS DAT /O EEN SHEET UIT PRINT ZOALS JE HET niet OP HET BEELDSCHERM ZIET. WAT IK HIERMEE BEDOEL IS : PROBEER HET ZELF MAAR EENS, DAT IS MAKKELIJKER DAN IK DIT PROBEER UIT TE LEGGEN (IK (DE VERTALER) HEB NOCH EEN PRINTER NOCH EEN ATOMCALC IN EPROM, DUS IK ZELF KAN HET NIET PROBEREN EN IK WEET DUS NIET ECHT HOE HET ER UIT ZAL ZIEN. HET VOORBEELD UIT DE HANDLEIDING IS NOU NIET BEPAALD EEN VOORBEELD VAN DUIDELIJKHEID).

STRING'S

Dit is een verhaal over de Atom string bewerkingen. Het is geen hoogdravend verhaal hoe de Atom dit even netjes in zijn slimme ROMmen oplost, doch gewoon een verhaal voor de gewone Atom man, dus voor de Atom basic programmeur.

Wat we eerst moeten weten is wat een string is. Een string is een bepaalde rij tekens met een bepaalde lengte afgesloten met als laatste teken een return.
Wat we verder moeten weten is dat de string ergens ligt opgeslagen. De Atom is de zelfde mening toegedaan en reserveert, indien gewenst (!) een stuk geheugen ruimte hiervoor. Dit reserveren van ruimte is uiteraard aan bepaalde voorwaarden gebonden:
1 Er moet voldoende geheugen geheugen voorradig zijn voor de opslag van de string. Dit lijkt misschien wat onlogisch doch het is het zelfde als: ik heb een blad met 66 lijnen. Op iedere lijn past 1 regel. Ik heb 72 regels om te schrijven. Past dit ja of nee. (en wie het niet gelooft probeert het maar!)
2 Men moet weten hoe lang de string maximaal kan worden. Hier wordt later nog op ingegaan.

Het plaats bepalen waar de string eventueel zou komen te staan gebeurt met een DIM statement. Dit DIM werkt alleen vanuit een programma. DIM bepaalt zijn ruimte op een leuke doch rechtlijnige manier. Om dit enigszins te verklaren is iets meer kennis van de Atom nodig. Zoals u wellicht weet moet een programma ergens beginnen. Dit wordt bijvoorbeeld door de Atom gedaan na een BREAK. Als u geen RAM hebt op #2900 dan wordt de wordt het programma neergezet op #8200. Waar een programma gaat beginnen ligt vast in een bepaalde waarde. Zo ook de textspace pointer. Dit is een duur woord voor een soort bladwijzer. Uw computer geheugen is dan het boek en de textspace pointer is de bladwijzer. Deze wijst dus naar het begin van het programma. Deze bladwijzer wijst dus iets aan. Daarom heet zo'n ding een pointer. Die pointer is ?18.

Terecht zult u denken als het begin vast ligt, dan zal het eind dus ook wel vast moeten liggen. Waar en niet waar. Een programma van 10 regels neemt beduidend minder plaats in dan een programma van 100 regels. (wie het niet gelooft: proberen maar!) In de Atom zit een soort teller dat het variabele eind van het programma in de smiezen houdt. Deze teller heet TOP. Als dus een regel wordt ingetikt met een nummer ervoor dan wordt de teller dus verhoogd, en wordt alleen een regelnummer met een return ingetikt dan wordt de waarde verlaagd. Wat gebeurt er nu na een DIM statement: De Atom kijkt waar de teller op staat (let op dat ik met opzet niet schrijf: kijkt waar het einde van het programma is) en reserveert een aantal bytes. Als u zegt DIM A(7) dan worder dus 7 bytes gereserveerd voor A. Als u nu nog een DIM geeft bij voorbeeld DIM B(8) dan wordt acher de ruimte die al voor A was gereserveerd, nog een stukje gereserveerd voor B.

En is het nu zo dat de gereserveerde ruimt vast ligt voor A en voor

B? Nee, was dat maar zo. Met een DIM legt u namelijk alleen vast waar $A of $B gaat beginnen. Als $A dus een lengte zou krijgen van 8 of meer, dan zit hij dus vrolijk het het werkgebied van $B zonder dat je een ERROR naar je hoofd geslingerd krijgt.

Zoals gezegd legt u dus de beginwaarde van de string vast. Deze wordt vastgelegd in de variabele die u opgeeft in de dim. Als u dus zegt DIM A(8),B(64) dan krijgt A bijvoorbeeld de waarde #29A0 en B de waarde #28A7. Dit houdt dus in dat u variabelen A en B niet meer in het programma mag gaan gebruiken want dan bent u het adres kwijt waar een eventuele $A zou moeten komen kwijt. Probeer maar eens:

```
10 REM STRING DEMO
20 DIM A(10),B(64)
30 PRINT &?18'
40 PRINT &TOP'
50 PRINT &A'
60 PRINT &B
70 END
```

Met DIM geeft u dus een waarde aan een variabele. Als A dus bij voorbeeld de waarde #2999 zou hebben en u tikt in in de commando mode $A="GEEN ASSEMBLER DOCH EEN TOP CURCUS", dan maakt het geen verschil of u intikt PRINT $A of PRINT $#2999. Het is dus keihard fout om het volgende te doen (en ik weet zeker dat ieder een zijn hoofd er al eens mee heeft gestoten):

```
10 REM FLUT DEMO
20 DIM A(17)
30 A=#1000
40 $A="FLUT TEST"
50 END
```

Een tweede punt waar we op moeten letten is dat de ruimte die voor de string wordt gereserveerd niet automatisch wordt schoongemaakt. Dat wil zeggen dat al u intikt DIM A(9), de ruimte voor de 9 posities voor $A vol met willekeurige rommel staat. Het volgende voorbeeld is dus gevaarlijk.

```
10 REM GEVAARLIJK DEMO
20 DIM A(9)
30 PRINT $A
40 END
```

De lengte van een string wordt bepaald met de instruktie LEN(expressie,of variable). De instruktie PRINT LEN(A) zoekt van af het begin van A naar een return, net zolang tot hij deze gevonden heeft en print deze. In plaats van A had er dus ook weer mogen staan #29A0 of iets dergelijks.

Iets anders wat we ook nog kunnen dimetioneren dat zijn zgn. Array's. Een array is een rij gegevens in de meest elementaire zin van het woord. Meestal hebben de elementen een onderlinge relatie bijvoorbeeld de temperaturen op eerste kerstdag 1983. Element 0 korrespondeert dan met tijdstip 0.00 en element 23 met het tijdstip 23.00

Ook hier geldt weer pas op met dimentioneren want als u zegt DIM AA(23) en u schrijft weg AA(25), weerderom weer geen error naar het hoofd geslingerd, doch wel fout, althans niet betrouwbaar. De standaard Atom rom controleerd de zgn. indices niet. Helaas. De verklaring uit de Atom manual is dat het tijd zou schelen. En inderdaad het scheelt veel tijd in het foutzoeken als je niet weet waar je het moet zoeken, maar dit terzijde. Gelukkig is er door een slimme vogel uit het hoge noorden een slimme P-CHARME uitgeknobbeld, die in ieder geval de indices bij multi-dimetionale arrays in de peiling houdt. Het dimetioneren van een array, bijvoorbeeld CC(100) heeft nu niets met de variabele C uit te staan CC(??) en C mogen dus onafhankelijk worden gebruikt.
Een simpel voorbeeldje voor het registreren van de temperaturen.

```
 10 REM TEMP DEMO
 20 REM A= HULPVARIABELE
 30 DIM AA(24)
 40 FOR N=0 TO 23
 50   PRINT "BRENG IN TIJDSTIP" N'
 60   INPUT A
 70   AA(N)=A
 80 NEXT N
 90 END
```

Wat aan dit voorbeeld opvalt is dat het niet is toegestaan om te schrijven INPUT AA(N). Dit resulteert in een dijk van een error.
Een andere methode is een gecombineerde array en string. Zoals u weet ligt het adres vast van een string. Dus kunnen we dit ook vast leggen in een array element. Probeer het volgende voorbeeld maar eenst te doorgronden.

```
 10 REM ARRAY STING DEMO
 20 DIM CC(100)
 30 FOR N=0 TO 10
 40   DIM B(10)
 50   CC(N)=B
 55   $CC(N)="ATOM"
 60 NEXT
 70
 80 FOR N=0 TO 10
 90   PRINT &CC(N)'
100 NEXT
110 END
```

Wellicht dat u nu het programma uit de manual bladzijde 62 nu ook beter begrijpt.
Wat uit dit verhaal duidelijk al zijn geworden is dat de waarde van TOP heel erg belangrijk is. Zorg dat deze goed staat. Als u een programma laadt met bijvoorbeeld *LOAD "DEMO", dan wordt top niet automatisch op de juiste waarde gezet. Sterker nog hij blijft staan op waar hij stond. Als deze dus bijvoorbeeld op #2902 staat em u begint een programma te runnen, dat een DIM in zich heeft, zeg dan maar baai baai zwaai zwaai tegen het programma want er wordt ruimte gereserveerd vanaf #2902. En daar stond nou net het programma. Geef dus altijd een END commando als je niet zeker weet of TOP goed staat.
Tot zover de strings en arrays voor deze keer.

65C02

Zoals iedereen waarschijnlijk al weet is het brein van onze ATOM een 6502. Vanaf de program counter wordt een byte de 6502 binnengehaald en behandeld als opcode. Afhankelijk van welke adresseringsvorm deze byte aangeeft zijn er een of twee of helemaal geen offsets nodig waarmee de 6502 verder kan. Er zijn dus principieel 256 verschillende opcodes mogelijk. Als we ATOMIC THEORY AND PRACTICE bekijken zien we dat lang niet alle mogelijke opcodes worden gebruikt. Als de 6502 tijdens het runnen een niet bestaande opcode tegenkomt dan zal de processor locken en kan men alleen met BREAK verder. Zoals ook al in ACORN NIEUWS nr1 jaargang 3 blz 22 te lezen valt zijn er sinds kort twee nieuwe tijpen 6502 op de markt n.l. de 65C02 en de R65C02. De R65C02 onderscheidt zich van de 65C02 door 4 extra instructies te weten BBS,BBR,SMB en RMB. De extra opcodes welke deze tijpen microprocessors begrijpen zijn eveneens in bovengenoemd artikel beschreven. Het zou eenvoudig zijn om een assembler programma te hebben dat op de ons bekende wijze deze opcodes genereert vanuit de source listing. Ik heb getracht dit te doen door de bestaande assembler in FROM aan te passen. Uiteindelijk wil ik de nieuwe assembler terug in FROM zetten hetgeen geen probleem oplevert als men de gewijzigde RESET FROM al heeft ingebouwd. Er wordt in de vele programmas die er voor de ATOM geschreven zijn niet of nauwelijks gebruik gemaakt van subroutines uit de assembler. Alleen #F291 wordt weleens gebruikt. Met de hierna gegeven assembler is het mogelijk om slechts de tabellen te verplaatsen zodat blijvend gebruik kan worden gemaakt van alle subroutines. Deze tabellen moeten worden verplaatst omdat de nieuwe assembler langer is dan de oude.

In het uitspitten van de oude assembler ben ik wat leuke annekdotes tegengekomen die misschien niet iedereeen al weet.

1) Achter alle opcodes met offset kan men tekst plaatsen zolang ; er niet in voorkomt. Dit heeft geen effect op het assembleren.

b.v.
```
PLA #80
RTSterug naar basic
LDA @#34 initialiseer letter
```
Het is niet nodig om eerst \ te geven voor de tekst. Bij het assembleren wordt de tekst wel op het scherm weergegeven.

2) Men kan i.p.v. [ook LINK #F2AC; geven zodat b.v. LINK#F2AC;PLA;RTS geldig wordt vanuit een basic programma.

3) Het is ook toegestaan om spaties te geven tussen de verschillende source code namen.

b.v.
```
L D A @#34 is hetzelde als
LDA @#34
```

4) Het is mogelijk om voor een label 2 letters te plaatsen zodat het iets duidelijker wordt wat een label betekent.

b.v.
```
:TO:LL34RTS
40TO:LL16BRK
```
Het is echter niet mogelijk om dit te doen in een offset b.v. BEQ TOLL34. Men kan echter wel na een ; of regelnummer steeds twee

letters plaatsen. Zo is ;TE;ST;RTS goed of 40TE;ST:LL3PLA;RTS eveneens.

5) Als men de assembler uit wil dan moet men dit natuurlijk doen met]. Als men echter eerst een spatie geeft dan ziet de assembler het] teken niet als einde assembler zodat een error optreedt als hierna een basic statement staat.
b.v. ;];PRINT "A" geeft een foutmelding.

6) Door de adressingsvormen immediate in accumulator b.v LDA@#34 en LSR A kunnen labels zoals @@ en AA niet worden gebruikt omdat de assembler ze ziet als adresserings vormen.

7) Als men een adres wil berekenn kan men (niet gebruiken omdat dit dan wordt gezien als een indirecte adresseringsvorm.

Om nu terug te komen op de 65C02 valt het op dat niet alle opcodes een logische waarde konden krijgen zodat de nieuwe assembler voor deze uitzonderingen een kleine aanpassing noodzakelijk was (regels 1400 - 1570). De nieuwe instructies en nieuwe adreseringsvormen worden in regels 1320 tot 1390 mogelijk gemaakt. Vanaf regel 1190 tot 1310 worden de oude tabellen verplaatst en het programma begint met een klein interpretertje waarvan de bedoelins is dat hij naar #1000 wordt geassembleerd. Dit kan ook naar #A000 maar dan moet op regel 1150 de JMP #A002 veranderd worden in JMP #C558. Men kan de nieuwe assembler bereiken door (te gebruiken en door) weer verlaten.

b.v.
```
10P=#2800
20(
30PHX;PLX;RTS
40)
50END
```

De betekenis van de verschillende nieuwe instructies stonden al in acorn nieuws, hier volgt slechts een lijst van alle nieuwe opcodes met de instructie erachter.

PHX	#DA	implied	ORA	#12	indirect	TRB	#1C	absolute
PHY	#5A	implied	SBC	#F2	indirect	TRB	#14	zero
PLX	#FA	implied	STA	#92	indirect	TSB	#0C	absolute
PLY	#7A	implied	ADC	#72	indirect	TSB	#04	zero
DEA	#3A	implied	AND	#32	indirect			
INA	#1A	implied	CMP	#D2	indirect			
BRA	#80	relatief	EOR	#52	indirect			
			LDA	#B2	indirect			
JMP	#7C	indirect,x	STZ	#9C	absolute			
BIT	#89	immediate	STZ	#64	zero			
BIT	#34	zero,x	STZ	#74	zero,x			
BIT	#3C	absolute,x	STZ	#9E	absolute,x			

De opcodes die geen logische waarde konden krijgen zijn:

```
JMP (#80,X)   geeft #40 moet zijn #7C
JMP (#2345)   geeft #51 moet zijn #6C
STZ #2345     geeft #6C moet zijn #9C
```

```
STZ #2345,X   geeft #7C moet zijn #9E
BIT @#34      geeft #20 moet zijn #89
```

Deze onlogische opcodes worden al in het assembler programma gecorrigeerd. Rest nog te vermelden dat als er nu een fout optreedt in een assembleer programma de gegeven error nummers anders zijn. Bij mij geeft

ERROR 76 foute opcode
ERROR 24 onjuiste adreseringsvorm

Ik denk dat op soortgelijke wijze de laatste 4 instructies bij te maken zijn, maar de R65C02 heb ik nog niet kunnen bemachtigen.

```
 10REM ********************************
 20REM **                            **
 30REM **  ASSEMBLER VOOR DE 65C02   **
 40REM **                            **
 50REM **                            **
 60REM **     DOOR  B.KASTEEL        **
 70REM **                            **
 80REM ********************************
 90REM JOSBOX IN VOET 3
100?#BFFF=#03;DIMLL(30);P.$12
110F.I=0TO30;LL(I)=-1;N.
120IN."WAAR ASSEMBLEREN "D
130?D=0;?(D+1)=0;P.$21
140F.I=0TO1
150P=D+2;[
160LDY#03;LDA(#05),Y;CMP@#3C;BEQ 16+A;JMP#A002
170:LL1;];N.;A=LL1;Q=A
180RELOC#F291,#F52B,A
190?(A+19)=#3E;?(A+174)=#3C
200P=A+305;[LDX@#4A;];P=A+309;[CMP 781+A,X;]
210P=A+318;[LDY 861+A,X;];P=A+325;[LDA 941+A,X;]
220P=A+330;[LDY 1021+A,X;];P=A+525;[LDA 1117+A,X;]
230P=A+534;[LDA 1101+A,X;];P=A+542;[LDA 1133+A,X;]
240P=A+549;[LDA 1149+A,X;]
250F.I=0TO#40
260?(A+781+I)=?(#F154+I);?(A+861+I)=?(#F194+I)
270?(A+941+I)=?(#F210+I);?(A+1021+I)=?(#F250+I)
280N.
290F.I=0TO#0E
300?(A+1101+I)=?(#F1E4+I);?(A+1117+I)=?(#F1D5+I)
310?(A+1133+I)=?(#F1F3+I);?(A+1149+I)=?(#F202+I)
320N.
330?(A+1130)=#C0;?(A+1143)=#04;?(A+1146)=#05;?(A+1124)=#03
340?(A+1112)=#20;?(A+1055)=#07
350!(A+846)=#8B8B8A8A;!(A+926)=#74727472
360!(A+1006)=#7AFA5ADA;!(A+1086)=#0D0D0D0D
```

```
370!(A+850)=#A51C5329;!(A+930)=#76C4C484
380!(A+1010)=#6C801A3A;!(A+1090)=#030C0D0D
390!(A+854)=#ADAC;!(A+934)=#06C6
400!(A+1014)=#0C1C;!(A+1094)=#0606
410F.I=0TO1
420P=A+667;Z=P;[LDX@#0A;JSR Q;CMP@#2C;BEQLL4;DEC#03;LDA#0F
430CMP@#0B;BEQLL8;CMP@#01;BEQLL5;CMP@#02;BEQLL5;JMP 521+A
440:LL8LDA#66;CMP@#4C;BNELL5;LDA@#67;STA#66
450:LL5LDX@#0D;JMP 522+A
460:LL4JSR Q;CMP@#59;BEQLL6;JMP 521+A::LL6JMP 522+A
470LDA#66;CMP@#2C;BNELL7;LDA@#95;STA#66
480:LL7JSR#C7BB;RTS
490LDX@#0B;LDA#66;CMP@#4C;BNELL9;LDA@#88;STA#66
500:LL9JMP 522+A
510LDX@#0C;LDA#66;CMP@#6C;BNE LL14;LDA@#9C;STA#66
520:LL14JMP#F49B
530LDY#25;BEQLL13;LDA#66;CMP@#6C;BNELL13;LDA@#8E;STA#66
540:LL13JSR Q;RTS;]
550N.;L=P
560P=A+499;[JMP 667+A;NOP;];P=A+451;[JSR 723+A;]
570P=A+493;[JMP 737+A;];P=A+447;[JMP 752+A;NOP;]
580P=A+382;[JSR 767+A;]
590?D=#40;?(D+1)=#BF;P.$6
600P."TOT #"&(#0F+1149+D)
610END
```

GEEN 200 K OP EEN SCHIJF.

Ik schrijf dit stukje, als reaktie op het artikel in 'ACORN NIEUWS 3,3' en wel om de volgende reden :

Een floppy disk heeft aan de binnenkant van het hoesje, een vilte beschermlaag zitten.
Dit vilt bestaat uit ragfijne haartjes, die in de draairichting staan, waardoor ze het schijfje schoonhouden.
Als wij nu zo'n schijf omdraaien, dan gaan we deze tegen de stand van de haartjes in draaien. Ik heb nu vier jaar ervaring met disk en vond op 'n bepaald moment ook deze verdubbelings methode. Alles ging goed totdat ik ontdekte dat er een aantal schijfjes beschadigd waren, en er geen enkel programma meer vanaf te krijgen was.
Dus men is gewaarschuwd voor de mogelijke gevolgen.

Het GFILL-statement van ROB van DORT versneld.

Omdat de machinetaal van het statement zo kort mogelijk moet worden gehouden, ben ik gaan kijken hoe dat bij GFILL mogelijk was. Het lijkt mij dat ik hierin geslaagd ben, aangezien ik de programma-lengte en de werktijd heb kunne halveren. (over het bijgeleverde test programma deed het oude statement 88 seconden en het nieuwe slechts 43 seconden)

De werkwijze van invullen van het figuur is hetzelfde gebleven.

```
   0 PROGRAM GFILL.STAT
  10 REMzie acorn-nieuws 3,2 (mei '84) pag.62
  20 DIMFF18;F.N=0TO18;FFN=-1;N.
  30 P.$21;P=A;GOS.a
  40 P.$6;P=A;GOS.a
  50 $T="GFILL";T=T+L.T
  60 ?T=FF0/256Ø#80;T?1=FF0%256;T?2=#80;T=T+2
  70 T!1=P;E.
  80aX=#57;Y=#52;B=#90;C=#91;S=#92;V=#93;W=#94
  85[
  90:FF0 JSR#C8BC;JSR#C231;JSR#C8BC;JSR#C4E4
 100       LDA#23,X;ORA#32,X;ORA#41,X;ORA#24,X;ORA#33,X;ORA#42,X
 110       BNEFF2
 120       LDA#14,X;STAX
 130       LDA#15,X;STAY
 140       LDA@0;STA#5B;STA#5D
 150       LDA@2;STA#5E;JSRFF17;STAB
 160       JSRFF18;CMPB;BEQFF4 \klaar als buiten scherm
 170      STAB;STY#5E;JSRFF18;CMPB;BNEFF1 \klaar als punt al geset
 180   INC#5E;JSRFF18;CMPB;BEQFF4 \klaar als kleur = achtergrond
 190       LDAB;STA(#5F),Y;INC#5E;BNEFF3
 200:FF1 LDAB;STA(#5F),Y;JMPFF4
 210:FF2 LDXS;BEQFF4;DEX;DEX;LDA#101,X;STAY
 220       LDA#100,X;STAX;STXS \pull X,Y
 230:FF3 LDA#B001;BMIFF6
 240:FF4 DEC#04;DEC#04;JMP#C55B \klaar
 250:FF5 INCX \ga naar rechts tot punt geset of buiten scherm
 260:FF6 LDAY;STA#5C;LDXX;INX;BEQFF7;STX#5A
 270       JSRFF18;STAB;JSRFF18;CMPB;BEQFF7
 280       EORB;AND(#5F),Y;BEQFF5
 290:FF7 LDA@1;STAV;STAW;BNEFF9
 300:FF8 LDAY;STA#5C;LDYX;BEQFF2;DEY;STY#5A
 310       JSRFF18;STAB \kijk of linkerkant leeg is
 320       JSRFF18;EORB;AND(#5F),Y;BNEFF2;DECX
 330:FF9 JSRFF17;INC#5E \kleur pixel
 340       LDYY;INY;BEQFF10;STY#5C
 350       JSRFF18;STAB \kijk boven
 360       JSRFF18;CMPB;BEQFF10;EORB;AND(#5F),Y;BEQFF11
 370:FF10INY
 380:FF11TYA;EORV;BEQFF12
 390       TYA;STAV;BNEFF12;LDYY;INY;JSRFF15 \push X,Y+1
 400:FF12LDAX;STA#5A;LDYY;BEQFF13;DEY;STY#5C \kijk onder
 410       JSRFF18;STAB;JSRFF18;EORB;AND(#5F),Y;BEQFF14
 420:FF13INY
 430:FF14TYA;EORW;BEQFF8
 440       TYA;STAW;BNEFF8;LDYY;DEY;JSRFF15 \push X,Y-1
 445       BNEFF8
```

```
450:FF15LDXS;LDAX;STA#100,X;TYA;STA#101,X;INX;INX \push X,Y
460 STXS;BNEFF16;PLA;PLA;JSR#FD1A;JSR#FD1A \stop want stack vol
470:FF16RTS
480:FF17LDAX;STA#5A;LDAY;STA#5C
490:FF18JMP(#3FE) \inkleur routine
500]
510 R.
```

Het test programma.

```
  0 REM DEMO GFILL
 10 GR.;!8=1;?12=0
 20 MOVE0,0;F.N=1TO50;PLOT6(A.R.%256)(A.R.%192);N.
 40 GFILL10,0;E.
```

En voor diegene die nog sneller willen, een programma dat alleen werkt in mode 4. Het is alleen niet geschreven als extra statement omdat ik niet in het bezit ben van P-CHARME. Je kunt erzelf een statement van maken, maar je kunt ook het startadres in regel 10 veranderen.
(Z=#xxxx;P=Z) en de routine dan aanroepen met:
X=A;Y=B;LINK Z.
waarbij A en B de cooordinaten van het startpunt zijn.
Deze routine doet een veld in maximaal 11 seconden (het test programma duurt nu nog maar 9 seconden)

```
  0 REM FILL IN MODE4
 10 DIMFF17;F.N=0TO17;FFN=-1;N.;F.N=1TO2;P=#800;P.$12$21
 20 Y=#5C;B=#C0;S=#C1;V=#C2;W=#C3;E
 30:FF0 STYY
 40      JSRFF16;BNEFF3;STYS
 50      AND(#5F),Y;BEQFF2;RTS
 60:FF1 LDYS;BEQFF3;DEY;DEY;LDA#101,Y;STAY;LDX#100,Y;STYS
 70:FF2 LDA#B001;BMIFF4
 80:FF3 RTS
 90:FF4 INX;BEQFF5;JSRFF16;AND(#5F),Y;BEQFF4
100:FF5 LDA@1;STAV;STAW;DEX
110:FF6 JSRFF16;STA#61;ORA(#5F),Y;STA(#5F),Y
120      LDYY;BEQFF7;LDY@32;LDA#61;AND(#5F),Y;BEQFF8
130:FF7 LDA@1
140:FF8 TAY;EORW;BEQFF9;TYA;STAW;BNEFF9;LDYY;DEY;JSRFF15
150:FF9 LDYY;CPY@191;BEQFF10;DEC#60
160      LDY@#E0;LDA(#5F),Y;INC#60;AND#61;BEQFF11
170:FF10LDA@1
180:FF11TAY;EORV;BEQFF12;TYA;STAV;BNEFF12;LDYY;INY;JSRFF15
190:FF12TXA;BEQFF14;DEX;TXA;AND@7;TAY;CPY@7;BNEFF13;DEC#5F
200:FF13LDA#F7C9,Y;LDY@0;AND(#5F),Y;BEQFF6
210:FF14JMPFF1
220:FF15TYA;LDYS;STA#101,Y;TXA;STA#100,Y;INY;INY;STYS;BNEFF17
230      PLA;PLA;JSR#FD1A;JMP#FD1A
240:FF16TXA;LSRA;LSRA;LSRA;STA#5F
250      LDA@#BF;SEC;SBCY;CMP@#C0;BCSFF0
260      LDY@0;STY#60;ASLA;ROL#60;ASLA;ROL#60
270      ASLA;ROL#60;ASLA;ROL#60;ASLA;ROL#60
280      ADC#5F;STA#5F;LDA#60;ADC@#80;STA#60;TXA
290      AND@7;TAY;LDA#F7C9,Y;LDY@0
300:FF17RTS
310]
320 P.$6;N.;E.
```

TALSTELSELS

Naar aanleiding van enkele vragen betreffende het hexadecimale talstelsel, heb ik een samenvatting gemaakt uit een artikel van W.G.E. Wouters over talstelsels.

1. Positionele talstelsels

Om een hoeveelheid elementen aan te duiden maken we, in het algemeen, gebruik van de cijfers: 0, 1, 2, 3, 4, 5, 6, 7, 8 en 9. Ook al bestaat deze hoeveelheid uit meer dan 'tien' elementen! Deze mogelijkheid wordt ons geboden doordat NIET alleen het cijfer, maar ook de plaats waar dit cijfer staat, bepalend is voor de getalwaarde die dat cijfer voorstelt. Met andere woorden: niet alleen het cijfer, maar ook de POSITIE, waar het cijfer staat, is van belang voor de bepaling van de getalwaarde. We noemen zo'n talstelsel een POSITIONEEL TALSTELSEL.

In een positioneel talstelsel bepalen twee zaken de getalwaarde:
- het cijfer
- de positie, die dat dat cijfer in dat getal inneemt

We geven van het bovenstaande een voorbeeld:
23123 = 20000 + 3000 + 100 + 20 + 3
Hoewel in het bovenstaande getal tweemaal het cijfer 2 en tweemaal het cijfer 3 voorkomt, is de getalwaarde van elk cijfer verschillend.

Opmerking:
Niet elk talstelsel is positioneel. Bij het turven van een getal wordt meestal gebruikt gemaakt van streepjes (▯).
Zo krijgen we de volgende getalaanduidingen:
1 = ▯ 2 = ▯▯ 3 = ▯▯▯ 4 = ▯▯▯▯ enz.
Ook het Romeinse talstelsel is niet zuiver positioneel:
I = 1 V = 5 X = 10 L = 50 C = 100 enz.
Zo is:
IV = 4 V = 5 VI = 6 VII = 7 VIII = 8 (niet positioneel!)
CCLXXXVII = 287 (niet positioneel!)

2. Het tientallige stelsel

In ons dagelijkse leven maken we gebruik van het tientallige stelsel, dat wil zeggen: we maken gebruik van tien speciale symbolen om getallen op te schrijven n.l. 0, 1, 2, 3, 4, 5, 6, 7, 8 en 9.
Het gebruik van deze tien symbolen heeft tot gevolg dat we na de 9 een COMBINATIE VAN CIJFERS moeten gaan gebruiken om een getalwaarde aan te geven. Zo krijgen we na de 9 de waarden 10, 11, 12, (enz.).
Omdat we met het tientallig stelel werken geldt ook het volgende:
62345 = 60000 + 2000 + 300 + 40 + 5

= 6*10000+2*1000+3*100+4*10+5*1
= 6*10↑4+2*10↑3+3*10↑2+4*10↑1+5*10↑0

Opmerking:
Wiskundig gezien is 10↑0 = 1
Zo geldt voor alle getallen a (a<>0): a↑0 = 1

Indien we de positie van elk cijfer vanaf de comma bepalen, dan zien we dat de positie van het cijfer, in het getal, de EXPONENT VAN HET GRONDTAL 10 aangeeft, waarmee we het cijfer moeten vermenigvuldigen om de getalwaarde van dit cijfer te bepalen.

Ook in decimale (comma) getallen bepaalt de positie van het cijfer, in het getal, de exponent van het grondtal 10, waarmee we het cijfer moeten vermenigvuldigen om de getalwaarde van dit cijfer te bepalen.
Zo is 154,234:
1*10↑2+5*10↑1+4*10↑0+2*10↑-1+3*10↑-2+4*10↑-3 =
1*100+5*10+4*1+2*0,1+3*0,01+4*0,001

Opmerking:
Wiskundig gezien geldt voor elke waarde -n: 10↑-n=1/10↑n

3. Het binaire stelsel

In het binaire of 2-tallige talstelsel kennen we alleen de cijfers: 0 en 1. Met behulp van deze cijfers kunnen we elke hoeveelheid een getalwaarde geven. De positie van elk cijfer is een exponent van het grondtal 2, waarmee we het cijfer moeten vermenigvuldigen om de decimale getalwaarde te bepalen. We geven hiervan een voorbeeld:
binair getal: 10011
decimaal: 1*2↑4+0*2↑3+0*2↑2+1*2↑1+1*2↑0 =
1*16+0*8+0*4+1*2+1*1 = 19

Omdat de computer de geheugenplaatsen in het interne geheugen met behulp van binaire getallen aanduidt (de adressen) kunnen we nu het volgende zeggen, voor een computer die werkt met 16 binaire cijfers (bits):
aantal bits: 16
aantal adressen (max.): 65536 (64K)
van 0 t/m 65535.

4. Het hexadecimale getalstelsel

Hoewel de computer alleen met binaire getallen werkt, schrijft men meestal deze binaire getallen in het 16-tallige stelsel (HEXADECIMAAL) op. De reden hiervoor is dat de binaire getallen meestal uit veel cijfers bestaan, en dat deze hierdoor moeilijk leesbaar zijn, waardoor men gemakkelijk fouten kan maken.
Het 16-tallige stelsel bezit ZESTIEN VERSCHILLENDE CIJFERS, dus we moeten zestien verschillende symbolen hebben om deze cijfers aan te duiden. Normaal gesproken worden deze als volgt genoteerd:

binair	decimaal	hexadecimaal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

Indien we met een hexadecimaal getal werken, dan zullen we dat aangeven door de letter H achter het getal te schrijven. Bij binaire getallen de letter B, en bij decimale de letter D.

5. Omzetting decimale naar binaire getallen

Deze omzetting gebeurt met de volgende methode:
we delen eerst het decimale getal door 2, de rest schrijven we achter de -- en de uitkomst delen we weer door 2 enz. De resten, die bij de delingen ontstaan zijn leveren ons het binaire getal. We lichten dit toe met enkele voorbeelden:

```
 16      13      23
2--0    2--1    2--1
  8       6      11
2--0    2--0    2--1
  4       3       5
2--0    2--1    2--1
  2       1       2
2--0    2--1    2--0
  1       0       1
2--1            2--1
  0               0
```

16D = 10000B 13D = 1101B 23D = 10111B

Let op: getallen, die in het decimale talstelsel niet repeteren, kunnen in een ander talstelsel, zoals het binaire, wel gaan repeteren!
We kunnen nu ELK decimaal getal terugbrengen tot een binair getal.

6. Omzetting van hexadecimale naar decimale getallen

```
25H = 2*16↑1+5*16↑0=2*16+5=37D
FFH = 15*16↑1+15*16↑0=240+15=255D
FFFF = 15*16↑3+15*16↑2+15*16↑1+15*16↑0
     = 61440+3840+240+15=65535D
```

7. Omzetting van decimale in hexadecimale getallen

Ook voor het omzetten van decimale in hexadecimale getallen kunnen we gebruik maken van de in punt 5 genoemde methode. We geven hiervan enkele voorbeelden:

```
      234              152
   16 --- 10D=AH  16 --- 8D=8H
       14                9
   16 --- 14D=EH  16 --- 9D=9H
        0                0

234D=EAH          152D=98H
```

8. Omzetting van binaire naar hexadecimale getallen

Deze laatste omzetting is de meest eenvoudige:

We verdelen het binaire getal in stukken van 4 bits, rechts te beginnen. Mochten er links geen 4 bits overblijven, dan worden deze aangevuld met nullen tot 4 bits. Van deze groepen schrijven we het hexadecimale equivalent op.

Enkele voorbeelden:

```
1011 0010B              dus: 10110010B = B2H
  B    2 H

1010 1111 0000 0100B    dus: 1010111100000100B = AF04H
  A    F    0    4 H
```

9. Omzetten van hexadecimale naar binaire getallen

Dit is eigenlijk de omgekeerde werkwijze als die in punt 8 reeds werd toegelicht.
Van elk hexadecimaal cijfer wordt het binaire equivalent opgeschreven. Bijvoorbeeld:

```
ACH = 1010 1100B
B80C = 1011 1000 0000 1100B
```

Ik hoop hiermee wat meer duidelijkheid gebracht te hebben in de eventuele verwarring betreffende enkele talstelsels.

Om uit de bestaande WORDPACK te komen is slechts een commando ter beschikking namelijk een "Q". Dit heeft als nadeel dat er vanaf #8200 een basicfile wordt gemaakt, wat als het een tekstfile betreft, waar de WORDPACK toch hoofdzakelijk voor wordt gebruikt, tot een serie foutmeldingen leidt en de machine vaak hangt. Het moeten gebruiken van de breaktoets heeft als nadeel dat #2900 en #2901 overschreven worden.
De oplossing is simpel. Door drie geheugenplaatsen te wijzigen wordt er door het aanslaan van de letter Q gewoon naar de basicruimte #82 gesprongen en een lees file aangemaakt (NEW). U kunt nu bevoorbeeld het wordpackfile op de MDCR zetten vanaf 2800 tot het eindadres dat U in de WORDPACK met het commando W heeft opgevraagd.

En dan nu de wijzigingen (hexadecimaal) :

geheugenplaats	was	wordt
ACF1	E1	94
ACF2	AA	FE
AAD9	15	07

Nu zijn op de WORDPACK de geheugenplaatsen van #AAE1 t/m #AB04 vrij gekomen.
Ik zelf gebruik die gedeeltelijk om iets te doen aan printersturing binnen een af te drukken regel. Normaal is het in de WORDPACK alleen mogelijk om de puntcommando's te laten beginnen achter een return. Ik gebruik nu binnen een regel de shift \. Het cijfer wat hierachter komt wordt net als bij .c als ASCII waarde naar de printer gestuurd.
Hierdoor is het bijvoorbeeld mogelijk om binnen een regel op schuin schrift over te stappen en weer terug. Dit werkt zelfs nog vlekkeloos bij het uitvullen van de regel.

Voor de mensen die hiermee toch willen experimenteren hier hoe ik dat doe. Allereerst in #AACC E1 zetten en in #AACD moet AA komen te staan. Dan vanaf #AAE1 de volgende codering leggen:

```
:LL0 CMP@#7C;BNE LL1
     JSR#A9DF
:LL1 JMP#FFF4
```

En zo simpel is dat.

ZEESLAG

```
 10 REM J.J. MERTENS
 20 REM         1983
 25 REM JOSBOX NOODZAKELIJK
 30 DIMAA20,BB4,V2,W3,P-1;T=0;P.$15;@=4
 40 U=#B002;P.$21;GOS.y;P.$6
 50 REM SHAPE
 60 REM SLAGSCHIP
 70 ?#2800=35
 80 !#2801=#44444444;!#2805=#65655665;!#2809=#55477457
 90 !#280D=#55555555;!#2811=#44440755;!#2815=#07555547
100 !#2819=#44444444;!#281D=#55554644;!#2821=#065555
110 REM KRUISER
120 ?#2828=22
130 !#2829=#65444444;!#282D=#40576565;!#2831=#55555575
140 !#2835=#64740755;!#2839=#44746474;!#283D=#4644
150 REM JAGER
160 ?#2844=13
170!#2845=#44075555;!#2849=#65444444;!#284D=#75075566;?#2851=#55
180 REM DUIKBOOT
190 ?#2858=7
200 !#2859=#44440755;!#285D=#355665
210 REM MIS
220 ?#2864=52
230 !#2865=#40242415;!#2869=#55537370;!#286D=#66151116
240 !#2871=#44422424;!#2875=#70400744;!#2879=#55537377
250 !#287D=#65515535;!#2881=#66561116;!#2885=#64E06060
260 !#2889=#40444424;!#288D=#7343474;!#2891=#35357777
270 !#2895=#55555335;
280 ?#289D=8;!#289E=#55333333;!#28A2=#50525225
290 TX.;P."         **  slagschip  **"'
300 P."JE MOET RAKEN:"'
310 P."  1 SLAGSCHIP   ;35 PTN/VELD"'
320 P."  2 KRUISERS    ;25 PTN/VELD"'
330 P."  3 JAGERS      ;20 PTN/VELD"'
340 P."  4 DUIKBOTEN   ;40 PTN/VELD"''
350 P."BIJ MEERDERE RAKETSCHOTEN ACHTERELKAAR TELT HET TWEEDE"'
360 P."DUBBEL, HET DERDE VOOR DRIE ETC."'
370 P."EEN GEMIST SCHOT KOST TWEE PTN, HET VOLGENDE VIER EN ZO"'
380 P."VERDER TOT 30 PTN."''
390 P."EVEN WACHTEN"
400 RESTORE
410 DATA1,7,5,3
420 DATA2,8,11,2
430 DATA3,9,17,1
440 REM 0T/M9
450 ?#3B00=10;!#3B01=#77175562;!#3B05=#66064473;!#3B09=#2535
460 ?#3B0E=7;!#3B0F=#77775762;!#3B13=#505500
470 ?#3B1C=8;!#3B1D=#73175562;!#3B21=#55754043
480 ?#3B2A=8;!#3B2B=#7175562;!#3B2F=#6447317
490 ?#3B38=8;!#3B39=#77575252;!#3B3D=#60442477
500 ?#3B46=10;!#3B47=#30535566;!#3B4B=#73175504;!#3B4F=#4044
510 ?#3B54=10;!#3B55=#34535562;!#3B59=#51357744;!#3B5D=#2066
520 ?#3B62=7;!#3B63=#7575526;!#3B67=#700707
530 ?#3B70=9;!#3B71=#34175562;!#3B75=#51357740;!#3B79=#66
540 ?#3B7E=10;!#3B7F=#77175562;!#3B83=#26424473;!#3B87=#5035
550 REM SLAGSCHIP
```

```
 560 B=6;D=1
 570 GOS.a;GOS.b;GOS.c
 580 REM¯KRUISERS,JAGERS,DUIKBTN
 590 F.N=1TO3;GOS.g;N.N
 600 P.$13"JE KUNT BEGINNEN"
 610 ?#80=135
 620 DO?#80=?#80-6;?#81=255-?#80;LI.BB0;U.?#80<6
 630 LI.#FFE3
 640 GR.;?#E1=0;P.$30
 650 MOVE20,4;DRAW210,4;DRAW210,154;DRAW20,154;DRAW20,3
 660 DRAW211,3;DRAW211,155;DRAW19,155;DRAW19,3
 670 F.P=1TO29;P." ";N.;P.$10$10$10$10$10
 680 P."s"$10$8"i"$10$8"a"$10$8"g"$10$8"s"$10$8"c"$10$8
 690 P."h"$10$8"i"$10$8"p"$30
 700 F.I=1TO10;MOVE(I*19+20),5;IFI<10 P=5
 710 IFI>9 P=4
 720 PLOTP,(I*19+20),155
 730 PLOT0,-10,7;SHAPE(#3B00+(I-1)*14);N.
 740 F.I=1TO10;MOVE20,(I*15+4);IFI<10 P=5
 750 IFI>9 P=4
 760 PLOTP,210,(I*15+4)
 770 PLOT0,-204,-7;SHAPE(#3B00+(10-I)*14)
 780 PLOT0,6,0;SHAPE#3B00;N.
 790 P=0;Q=0;R=0;S=0
 800 REM¯ZOEKEN
 810iIFP>=20 G.l
 820 P.$30"SCORE "S"  MAX SCORE "T
 830 ON ERR P.$11$11;PLAYG2.;G.840
 840 P.'"COORDINATEN      "$8$8$8$8$8;IN.N
 850 IFN<0ORN>100 P.$11$11;PLAYG#4.;G.840
 860 X=N%10*19+30;Y=(9-N/10)*15+11
 870 F.O=1TO20
 880 IFAAO%100=N;?21=?21-1;G.j
 890 N.O;MOVEX,Y;SHAPE#2864;SHAPE#289D;Q=Q+1;R=0
 900 ?#80=130;?#81=45;LINKBB0
 910 IFQ>15 Q=15
 920 S=S-2*Q;G.i
 930 REM¯RAAK
 940jIFAAO/100=1 G.k
 950 ?#80=12;?#81=1;LI.BB3;?#B002=1
 960 Q=0;R=R+1;AAO=AAO+100;P=P+1
 970 IFO<5 S=S+R*35;MOVEX,Y;SHAPE#2800;G.i
 980 IFO<11 S=S+R*25;MOVEX,Y;SHAPE#2828;G.i
 990 IFO<17 S=S+R*20;MOVEX,Y;SHAPE#2844;G.i
1000 S=S+R*40;MOVEX,Y;SHAPE#2858;G.i
1010kGOS.z
1020 P.$30"SUKKEL, DIE HEB JE AL GEHAD"'"DIT KOST JE 10 PTN."
1030 ?#80=150
1040 DO?#80=?#80-3;?#81=255-?#80;LI.BB0;U.?#80<5
1050 GOS.z
1060 S=S-10;R=0;G.i
1070 REM¯ALLE SCHEPEN GEVONDEN
1080lPLAYD"8.,E"8.,F"8.,G"8.
1090 GOS.z
1100 P.$30"SCORE "S"  MAX SCORE"T
1110 IFS<0 IFS<T;T=S
1120 IFS>T;T=S
1130 IN.'"NOG EEN SPEL"$W
1140 IF?W=CH"N" P.$12;END
1150 P.$12;IN."WIL JE INSTRUCTIES"$W
1160 IF?W=CH"J" G.290
```

```
1170 G.390
1180aREM¯HOR/VERT
1190 A=A.R.%40;IFA<20 A=1;R.
1200 A=10;R.
1210bREM¯STARTCOORD
1220 X=A.R.%(B+1);Y=A.R.%(B+1)
1230 Z=X+10*Y
1240 R.
1250cREM¯LEGGEN
1260 F.C=B TO9
1270 AAD=Z;Z=Z+A;D=D+1
1280P.$21
1290 P.D-1,AA(D-1)
1300P.$6
1310 N.C
1320 R.
1330dREM¯DUBBELTEST
1340 K=0
1350 F.I=1TOD-1
1360 F.J=D TO(D+9-B)
1370 H=A.(AAI-AAJ)
1380 IF1>=H G.f
1390 IF8>=H G.e
1400 IF11>=H G.f
1410eN.J;N.I;R.
1420fK=1;?21=?21-2;R.
1430gREM¯VOLTOOI
1440 READG,M,F,L
1450hB=M;D=F
1460 GOS.a;GOS.b;GOS.c
1470 B=M;D=F;GOS.d
1480 IFK G.h
1490 IFG G=G-1;F=F+L;G.h
1500 R.
1510zP.$30"                                                  "
1520 P."                                                     ";R.
1530y[;:BB0LDAU;LDY#81;:BB1LDX#80;:BB2DEX;BNEBB2;EOR@4
1540STAU;DEY;BNEBB1;RTS
1550:BB3LDY#81;LDX#80;:BB4LDAV;STAU;AND@#48;ADC@#38;ASLA;ASLA
1560ROLV+2;ROLV+1;ROLV;DEY;BNEBB4;DEX;BNEBB4;RTS
1570];R.
```

In machinetaal programmeren het error getal welke onze ATOM produceert 2 bytes hoger is dan de lage byte van de program counter waar de BRK instructie optrad.
b.v. Als er een BRK (#00) op #2809 staat dan zal bij LINK#2809 error #09+2=11 ontstaan.

Wil men in een machinetaal programma een bepaalde error nummer laten optreden dan kan men met de standaard error handler LDA@#34;PHA;PHA;JMP#C9D8 uitvoeren of LDA@#34;JMP#C9DA, waarbij het error nummer dan #34 (=52) is.
Met het ASBK-S systeem begint de error handler op LL16 = #1105. Het error getal zit dan in geheugen plaats #AF, zodat met LDA@#34;STA#AF;JMP#1105 een error boodschap nieuwe stijl wordt verkregen.

Als men een basic programma wil laden met *RUN dan gaat het mis als het programma een DIM statement bevat. De reden hiervoor is dat de pointer voor het dimensioneren nog niet goed is gezet als men het DIM statement tegenkomt. Dit kan worden voorkomen door in het begin van het programma de regel !#0D=#XXXX;?#23=TOP&#FF;?#24=TOP/256&#FF op te nemen, waarbij #XXXX de TOP van het programma is of een geschikt geheugengebied om label adressen op te slaan.

In een machinetaal programma u een tekst kan laten afdrukken met een routine welke op #F7D1 begint. De print routine wordt beeindigd als er een karakter wordt tegengekomen waarvan de highbyte geset is. (dus opcode =#80 of hoger)
b.v.

```
JSR#F7D1;]
$P="HALLO";P=P+L.P
[;NOP;RTS
etc.
```

NOP heeft een opcode van #EA zodat hiermee de printroutine wordt beeindigd en wordt de NOP instructie uitgevoerd alsof het een normale opcode betreft.

Men een basic programma kan laten runnen welke niet op een 256 bytes grens begint door eerst #05 en #06 goed te zetten en dan naar #C2F2 te springen.
b.v.

```
10A=#2823
20$A="P.A;END";A=A+L.A
30?A=#0D;A=A+1
40?A=#FF;A=A+1
50P=#2800;[
60LDA@#23;STA#05
70LDA@#28;STA#06
80JMP#C2F2;]
90LINK#2800;END
```

Met XDUMP #2823 ziet u waar het programma staat.

STATEMENTS

Uit de regio Den Haag hebben wij diverse statements uit de JOSBOX ontvangen. Deze zijn in source omgezet om in een eigen box te gebruiken of samen met P-CHARME. Zij volgen hier zonder verder commentaar.

```
 10REM GRMODE
 20J=60;DIMLLJ;F.I=0TOJ;LLI=#FFF;N.
 30LL43=Z+#301;LL44=Z+#30A;LL45=Z+#313
 40P.$21;P=A;Z=P;GOSUB a
 50P.$06;P=A;Z=P;GOSUB a
 60 $T="GRMOD";T=T+LENT
 65 ?T=(Z+#206)/2560#80;T?1=(Z+#206)%256;T=T+2
 70 $T="TXMOD";T=T+LENT
 75 ?T=(Z+#202)/2560#80;T?1=(Z+#202)%256;T?2=#80;T=T+2
 80 A=Z+#3FF;T!1=A
 90 END
100a;[
110:LL0 BRK;BRK;BRK;BRK;BRK
120JSRLL7;JMPLL21;JSRLL10;JMPLL21;JSRLL9
130JMPLL21;LDA@#F0;STA#B000;LDA@#80;STA#E0
140ASLA;STA#5F;TAY;LDX@#18
150:LL3 STA(#5F),Y;INY;BNELL3;INC#60;DEX
160BNELL3;DEY;STY#E1;LDA@#00;STA#DF
170STA#E0;JMPLL19;CLC;LDX@#10;STX#E6
180LDX@#08;JSR#FD13;JMPLL21
190:LL4 CLC
200:LL5 JMP#FD11
210:LL6 JSR#FD1A;JMPLL21
220:LL7 DEC#E0;BPLLL8;INC#E0;JSRLL9;BEQLL8
230STA#E0
240:LL8 RTS
250:LL9 LDA#DF;BEQLL8;DEC#DF;LDA@#1F;RTS
260:LL10 LDA#DF;INC#DF;CMP@#0F;BNELL16;LDY#E6
270BMILL12;DEY;BNELL12
280:LL11 JSR#FE71;BCSLL11;LDY@#10
290:LL12 STY#E6;DEC#DF;LDA@#80;STA#60;STA#5F
300STA#E3;LDY@#00;STY#E2;LDX@#17
310:LL13 INC#60
320:LL14 LDA(#5F),Y;STA(#E2),Y;INY;BNELL14;INC#E3
330DEX;BNELL13;TYA;DEC#60
340:LL15 STA(#E2),Y;STA(#5F),Y;INY;BNELL15
350:LL16 RTS
360:LL17 CMP@#07;BEQLL6;LDX@#09;JSR#FEC5;BNELL21
370LDALL43,X;PHA;LDALL44,X;PHA;RTS
380:LL18 CMP@#06;BEQLL4;CMP@#15;BEQLL5;LDY#E0
390BMILL16;JSRLL21;CMP@#20;BCCLL17;CMP@#7F
400BEQLL23;JSRLL24;LDY#E0;INY;CPY@#20
410BCCLL20;JSRLL10
420:LL19 LDY@#00
430:LL20 STY#E0
440:LL21 PHA;JSRLL29;LDX@#02;STY#DE
450:LL22 LDA(#5F),Y;EOR#E1;STA(#5F),Y;TYA;CLC
460ADC@#20;TAY;CPY#DE;BNELL22;ROR#DE
```

```
470INC#60;DEX;BNELL22;PLA;RTS
480:LL23 JSRLL7;LDA@#20;JSRLL24;BMILL21
490:LL24 AND@#7F;LDX@#6D;CMP@#40;BCCLL25;INX
500:LL25 STX#E3;TAX;AND@#1F;STA#E2;LDY@#FF
510CPX@#60;BCSLL26;CPX@#20;BCCLL26;INY
520:LL26 STY#DE;JSRLL29;LDY@#40;JSRLL28;LDA#5F
530ORA@#40;STA#5F;LDY@#20
540:LL27 LDA(#E2),Y;EOR#DE;STA(#5F),Y;TYA;CLC
550ADC@#20;TAY;BNELL27;INC#60;LDY@#20
560:LL28 LDA#DE;STA(#5F),Y;TYA;SEC;SBC@#20
570TAY;BPLLL28;RTS
580:LL29 LDA#E0;ASLA;STA#5F;LDA#DF;ASLA
590ADC#DF;SEC;RORA;STA#60;ROR#5F
600LDY@#00;RTS;JSR#FEFB;PHP;PHA
610CLD;STX#E4;STY#E5;JSRLL18;JMP#FE5F
620PHP;CLD;STX#E4;STY#E5
630:LL30 BIT#B002;BVCLL31;JSR#FE71;BCCLL30
640:LL31 JSR#FB8A
650:LL32 JSR#FE71;BCSLL32;JSR#FE71;BCSLL32;TYA
660CMP@#06;BEQLL33;CMP@#07;BEQLL33;CMP@#0E
670BEQLL34;JMP#FEB2
680:LL33 AND@#05;ROL#B001;ROLA;JSRLL18;JMPLL30
690:LL34 LDA@#6D;JSRLL36;LDA@#6E;JSRLL36;TYA
700:LL35 JMP#FE60
710:LL36 STA#E3;LDX@#FF
720:LL37 STX#DE;JSRLL29;LDA#5F;ORA@#40;STA#5F
730:LL38 STY#E2;LDY@#20
740:LL39 LDA(#5F),Y;EOR#DE;EOR#E1;CMP(#E2),Y;BEQLL40
750LDY#E2;INY;CPY@#20;BNELL38;INX
760BEQLL37;RTS
770:LL40 TYA;ADC@#1F;TAY;BNELL39;PLA
780PLA;LDA#E2;STY#E2;LDY#DE;INY
790ORA(#E2),Y;BNELL35;];?P=#80;P?1=#20;P=P+2;[
800LDY@#04;BNELL41;LDY@#08
810:LL41 LDX@#03
820:LL42 LDALL45,Y;STA#208,X;DEY;DEX;BPLLL42;INX;STX#E1
830LDA@#0C;JSR#FFF4;JMP#F6AD;BRK
840]
850B=Z+#220;E=Z+#3FF
860B!0=#1800;B!2=#3636;B!4=#6018;B!6=#C38;B!8=#300C
870B!10=#0;B!12=#0;B!14=#0;B!16=#183C;B!18=#3C3C
880B!20=#7E0C;B!22=#7E1C;B!24=#3C3C;B!26=#0;B!28=#C
890B!30=#3C30;B!32=#1800;B!34=#3636;B!36=#663E;B!38=#186C
900B!40=#1818;B!42=#1818;B!44=#0;B!46=#600;B!48=#3866
910B!50=#6666;B!52=#601C;B!54=#630;B!56=#6666;B!58=#3838
920B!60=#18;B!62=#6618;B!64=#1800;B!66=#7F36;B!68=#C58
930B!70=#306C;B!72=#C30;B!74=#185A;B!76=#0;B!78=#C00
940B!80=#186E;B!82=#606;B!84=#7C3C;B!86=#C60;B!88=#6666
950B!90=#3838;B!92=#7E30;B!94=#60C;B!96=#1800;B!98=#3600
960B!100=#183C;B!102=#38;B!104=#C30;B!106=#7E3C;B!108=#7E00
970B!110=#1800;B!112=#187E;B!114=#1C0C;B!116=#66C;B!118=#187C
980B!120=#3E3C;B!122=#0;B!124=#60;B!126=#C06;B!128=#1800
990B!130=#7F00;B!132=#301A;B!134=#6D;B!136=#C30;B!138=#183C
1000B!140=#38;B!142=#3000;B!144=#1876;B!146=#618;B!148=#67E
1010B!150=#3066;B!152=#666;B!154=#3838;B!156=#7E30;B!158=#180C
1020B!160=#0;B!162=#3600;B!164=#667C;B!166=#66;B!168=#1818
```

```
1030B!170=#185A;B!172=#38;B!174=#6038;B!176=#1866;B!178=#6630
1040B!180=#660C;B!182=#3066;B!184=#C66;B!186=#3838;B!188=#18
1050B!190=#18;B!192=#1800;B!194=#3600;B!196=#618;B!198=#3B
1060B!200=#300C;B!202=#18;B!204=#70;B!206=#38;B!208=#7E3C
1070B!210=#3C7E;B!212=#3C0C;B!214=#303C;B!216=#383C;B!218=#7000
1080B!220=#C;B!222=#1830;B!224=#4060;B!226=#6B6B;B!228=#6B6B
1090B!230=#6B6B;B!232=#6B6B;B!234=#46B;B!236=#AD8;B!238=#1610
1100B!240=#3BE2;B!242=#3240;B!244=#FE52;B!246=#FE94;B!248=#6C6E
1110B!250=#6C7E;B!252=#4553;B!254=#D57;B!256=#3C3C;B!258=#3C7C
1120B!260=#7E78;B!262=#3C7E;B!264=#7E66;B!266=#6606;B!268=#6360
1130B!270=#3C66;B!272=#3C7C;B!274=#3C7C;B!276=#667E;B!278=#6366
1140B!280=#6666;B!282=#7C7E;B!284=#3E00;B!286=#18;B!288=#6666
1150B!290=#6666;B!292=#606C;B!294=#6660;B!296=#1866;B!298=#6C06
1160B!300=#7760;B!302=#6666;B!304=#6666;B!306=#6666;B!308=#6618
1170B!310=#6366;B!312=#6666;B!314=#6006;B!316=#660;B!318=#183C
1180B!320=#666E;B!322=#6066;B!324=#6066;B!326=#6060;B!328=#1866
1190B!330=#7806;B!332=#7F60;B!334=#6676;B!336=#6666;B!338=#6066
1200B!340=#6618;B!342=#6B66;B!344=#663C;B!346=#600C;B!348=#630
1210B!350=#3066;B!352=#7E6A;B!354=#607C;B!356=#7C66;B!358=#6E7C
1220B!360=#187E;B!362=#7006;B!364=#6B60;B!366=#667E;B!368=#667C
1230B!370=#3C7C;B!372=#6618;B!374=#6B66;B!376=#3C18;B!378=#6018
1240B!380=#618;B!382=#7E00;B!384=#666E;B!386=#6066;B!388=#6066
1250B!390=#6660;B!392=#1866;B!394=#7806;B!396=#6B60;B!398=#666E
1260B!400=#6A60;B!402=#66C;B!404=#6618;B!406=#7F66;B!408=#183C
1270B!410=#6030;B!412=#60C;B!414=#3000;B!416=#6660;B!418=#6666
1280B!420=#606C;B!422=#6660;B!424=#1866;B!426=#6C66;B!428=#6360
1290B!430=#6666;B!432=#6C60;B!434=#6666;B!436=#6618;B!438=#773C
1300B!440=#1866;B!442=#6060;B!444=#606;B!446=#1800;B!448=#663C
1310B!450=#3C7C;B!452=#7E78;B!454=#3C60;B!456=#7E66;B!458=#663C
1320B!460=#637E;B!462=#3C66;B!464=#3660;B!466=#3C66;B!468=#3C18
1330B!470=#6318;B!472=#1866;B!474=#7C7E;B!476=#3E00;B!478=#0
1340B!480=#FF00
1350F.L=(Z+#302)TO(Z+#30A);?L=Z/256;N.
1360?(Z+#319)=Z/256+1;?(Z+#31B)=Z/256+1;?(Z+#114)=Z/256+2
1370?(Z+#1B8)=Z/256+2;?(Z+#1BD)=Z/256+3
1380RETURN
```

```
  10REM PLAY
  20J=20;DIMLLJ;F.I=0TOJ;LLI=#FFF;N.
  30 P.$21;P=A;GOSUB a
  40 P.$6;P=A;GOSUB a
  50 $T="PLAY";T=T+LEN(T)
  60 ?T=LL0/2560#80;T?1=LL0%256;T?2=#80;T=T+2
  70 A=P;T!1=A;END
  80P.$21;F.I=1TO2;P=Z
  90a;[
 100; :LL21INY
 110; :LL1;LDA(#05),Y;CMP@#20;BEQ LL21;STY #03;RTS
 120; :LL0; ;JSRLL1;CMP@#52;BNELL2;INY;STA#54
 130BNELL8
 140; :LL2; ;CMP@#41;BCCLL4+1;CMP@#48;BCSLL4+1;STA#54
 150INY;LDA(#05),Y;CMP@#22;BNELL3;LDA@#1C
 160BNELL6
 170; :LL3; ;CMP@#27;BNELL4;LDA@#0E;BNELL6
 180; :LL4; ;LDA@#00;DEY
 190; :LL6; ;INY;CLC;ADC#54;STA#54;LDA(#05),Y
```

```
200CMP@#23;BNELL7;LDA#54;ADC@#06;STA#54
210INY
220;:LL7;;LDX#54;LDALL12,X;STA#54;LDA@#05
230;:LL8;;STA#92;LDX@#03;LDA(#05),Y;SEC;SBC@#30
240INX
250;:LL9;;DEX;BMILL4+1;CMP#F7CD,X;BNELL9;LDALL17,X
260STA#53;INY;LDA(#05),Y;CMP@#2E;BNELL10
270LDA#53;LSRA;ADC#53;STA#53;INY
280;:LL10;;STY#03;LDY@#00;STY#52;STY#93;SEC
290;:LL11;;INY;BNELL12;INC#93
300;:LL12;;LDA#52;SBC#54;STA#52;LDA#53;SBC@#00
310STA#53;BCSLL11;TYA;LDX@#03
320;:LL13;;ASLA;ROL#93;DEX;BNELL13;TAY
330LDA@#00;INY;INC#93
340;:LL14;;EOR#92;STA#B002;LDX#54
350;:LL15;;DEX;NOP;NOP;NOP;BNELL15
360DEY;BNELL14;DEC#93;BNELL14;LDY#03
370JSRLL1;CMP@#2C;BNELL16;INY;JMPLL0
380;:LL16;;JMP#C558
390];LL17=P+#2A
400P!0=#D9F6;P!2=#B7CD;P!4=#9AA3;P!6=#E789;P!8=#C2CD
410P!10=#9AAD;P!12=#8191;P!14=#6C79;P!16=#5B66;P!18=#4C51
420P!20=#7244;P!22=#6066;P!24=#4C56;P!26=#4048;P!28=#363C
430P!30=#2D32;P!32=#2527;P!34=#3921;P!36=#2F32;P!38=#2529
440P!40=#1F23;P!42=#804;P!44=#2010;P=P+46
450RETURN
```

```
10 REM FIND
30J=22;DIMLLJ;F.I=0TOJ;LLI=#FFF;N.
40 P.$21;P=A;GOSUB a
50 P.$6;P=A;GOSUB a
60 $T="FIND";T=T+LEN(T)
70 ?T=LL0/256@#80;T?1=LL0%256;T?2=#80;T=T+2
80 A=P;T!1=A
90 END
100
110a;[
115:LL3LDA6;CMP@1;BNEP+3;LDX@0;STX4;RTS
120;:LL0;;JSRLL3;JSR#CEFA;STY#34;STY#43;STY#58
130STA#59;LDA#53;CMP@#01;BNELL4;LDA#52
140CMP@#40;BEQLL6
150;:LL4;;DEY
160;:LL5;;INY;LDA(#52),Y;STA#140,Y;CMP@#0D;BNELL5
170;:LL6;;LDY@#FF
180;:LL7;;LDX@#00
190;:LL8;;INY
200;:LL9;;LDA(#58),Y;CMP#140,X;BEQLL10;CMP@#0D;BNELL8
210INY;LDA(#58),Y;BMILL13;STA#25;INY
220LDA(#58),Y;STA#16;JSR#CEA1;BNELL9
230;:LL10;;STY#57
240;:LL11;;INX;INY;LDA#140,X;CMP@#0D;BNELL12
250JSRLL14;BNELL7
260;:LL12;;CMP(#58),Y;BEQLL11;LDY#57;BNELL7
270;:LL13;;JMP#C55B
280;:LL14;;LDA#321;PHA;LDA@#05;STA#321;JSR#C589
290PLA;STA#321;LDY@#01
300;:LL15;;LDA(#58),Y;CMP@#0D;BEQLL16;JSR#CA4C;INY
310BNELL15
320;:LL16;;JSR#CD54;DEY;RTS
330];RETURN
```

Search is een statement om een string in het geheugen op te zoeken. De syntax luidt: SEARCH (string).

```
 10REM *SEARCH
 20REM BEDOELD ALS STATEMENT
 30REM OM STRING OP TE ZOEKEN
 40REM IN HET GEHEUGEN
 50REM BV.IN EIGEN INTERPRETER
 60REM F.G. LE BLANC - MONSTER
 70REM TEL 01749-42635
 80REM #61 BYTES
 90G=#10
100DIMLL11;F.N=0TO11;LLN=-1;N.
110P.$21;P=A;GOS.a
120P.$6;P=A;GOS.a
130$T="SEARCH";T=T+LENT
140?T=LL11/256\#80;T?1=LL11%256;T?2=#80;T=T+2
150A=P;T?1=A
160END
170a;[;:LL11
180DEY
190:LL0INY
200LDA(5),Y;CMP@#20;BEQLL0
210CMP@#22;BEQLL1;RTS
220:LL1INY;JSR#CEBF
230LDA@#0;STA#80;LDA@G;STA#81
240LDY@#FF
250:LL2
260LDX@0
270STX#82
280DEX
290:LL3
300INX
310:LL4
320INY
330CPY@0;BNELL6
340INC#81
350LDA@#0;CMP#81
360BNELL6;JMP#C558
370:LL6
380LDA#140,X
390CMP@#0D;BEQLL8
400CMP(#80),Y;BNELL2
410LDA@0;CMP#82
420BNELL7;STY#82
430:LL7
440JMPLL3
450:LL8
460LDA@#23;JSR#FFF4
470JSR LL9;JSR#FFED
480INY;JMP LL2
490:LL9
500LDX@0
510:LL10
520LDA#81,X
530JSR#F802
540INX
550CPX@2;BNELL10
560RTS
570]:RETURN
```

Hierbij een listing met twee handige statements nl. APPEND en VERIFY.

APPEND :
========

Dit commando geeft de mogelijkheid om twee BASIC-programma's aan elkaar te koppelen. Eerst het eerste programma in laden.(zorg er voor dat de page-pointer(?18) goed staat) Daarna het tweede programma inladen via :

APPEND"PROGRAMMA NAAM"

en het tweede programma wordt dan direkt achter het eerste geladen.

VERIFY :
========

Dit commando geeft de mogelijkhed om te contoleren of een progamma goed is gesaved. Spoel hiertoe de band naar het begin van het programma terug en dan :

VERIFY"PROGRAMMA NAAM"

Beide programma's kunnen zowel met als zonder naam gebruikt worden.(dus zoals bij LOAD)
Bij VERIFY kunnen twee foutmeldingen optreden :

66 Verschil tussen band en geheugen in een naamloos programma.

130 Idem, maar dan voor een programma met naam.

Men moet oppassen dat tijdens he verifyen het beeldscherm niet scrollt, want dan worden er een paar bits gemist. Het is ook mogelijk om een gedeelte van een programma te verifieren. Dus als er door scrollen een 'ERROR' optreed, dan kan de band een stukje terug gespoeld en het verifieren voortgezet worden.

```
  10 PROGRAM APPEND & VERIFY
  20
  30 DIMLL10;F.I=0TO10;LLI=999;N.
  40 P.$21;P=A;GOS.a
  50 P.$6 ;P=A;GOS.a
  60 $T="APPEND";T=T+L.T
  70 ?T=LL0/256Ø#80;T?1=LL0%256;T=T+2
  80 $T="VERIFY";T=T+L.T
  90 ?T=LL1/256Ø#80;T?1=LL1%256;T?2=#80;T=T+2
 100 A=P;T!1=A
 110 END
 120
 130a[
 140\append-commando
 150:LL0;JSR #F818
 160       SEC
 170       LDA #0D;SBC @#02;STA #CB
 180       LDA #0E;SBC @#00;STA #CC
 190       JSR #F962;JMP #CD9B
```

```
200\verify-commando
210:LL1;JSR #F818
220      LDX@#C9;JSR #F84F
230      PHP;JSR #FC3E
240      PLP;BNE LL7
250:LL2;JSR #FB8E;BVC LL6;BEQ LL2
260      JSR #FC2F
270      LDY@#00
280:LL3;JSR #FFD4
290      CMP(#CB),Y;BEQ LL4
300      BRK
310:LL4;INC #CB;BNE LL5
320      INC #CC
330:LL5;LDX@#D4;JSR #FA08;BNE LL3
340:LL6;JMP #CD9B
350:LL7;JSR #FB8E;BVCLL6;BNELL7
360      JSR #FBC9;BNELL7
370      JSR #FBE2;JSR #F992;JSR #F7EC
380      BIT #DB;BVC LL8
390      INX
400      JSR #F7F1
410      LDA #FD,X;JSR #F802
420:LL8;JSR #FFED
430      LDY@#FF
440:LL9;INY
450      JSR #FFD4
460      CMP(#D4),Y;BEQ LL10
470      BRK
480:LL10CPY #D8;BNELL9
490      LDY #DB;BMILL7
500      JMP #CD9B
510];RETURN
```

Met onderstaande routine kan men in totaal 9 schermen in het geheugen bewaren en terughalen.
Het statement wordt gevolgt door het schermnummer.

```
 10 PROGRAM SCREEN-GSCREEN
 15 \ NU OOK VAN UIT EEN PROGRAMMA TE GEBRUIKEN!!!!
 20
 30 DIMLL8;F.I=0TO8;LLI=999;N.
 40 P.$21;P=A;GOS.a
 50 P.$6 ;P=A;GOS.a
 60 $T="SCREEN";T=T+L.T
 70 ?T=LL0/256@#80;T?1=LL0%256;T=T+2
 80 $T="GSCREEN";T=T+L.T
 90 ?T=LL3/256@#80;T?1=LL3%256;T=T+2;?T=#80
100 A=P;T!1=A
110 END
120
130a[
140:LL0JSRLL6;LDY@#00;STY#C0
150:LL1DEY;LDA#8000,Y;STA(#C0),Y;CPY@#00;BNELL1
160INC#C1
170:LL2DEY;LDA#8100,Y;STA(#C0),Y;CPY@#00;BNELL2
180JMP#C55B
190:LL3JSRLL6;LDY@#00
```

```
200:LL4DEY;LDA(#C0),Y;STA#8000,Y;CPY@#00;BNELL4
210INC#C1
220:LL5DEY;LDA(#C0),Y;STA#8100,Y;CPY@#00;BNELL5
230JMP#C55B
240:LL6JSR#C4E1;LDX#4;DEX;STX4;LDA#16,X;CMP@#10;BPLLL7
245\ CMP@#09 IS HET AANTAL SCHERMEN
250CMP@1;BMILL7;TAX;LDA@#80
260:LL8CLC;ADC@2;DEX;CPX@0;BNELL8;STA#C1;RTS
270:LL7LDA@#F2;PHA;PHA;JMP(#202)
280]
290RETURN
```

```
 10 PROGRAM DSIZE
 20 DIM LL2;FOR I=0 TO 2;LLI=#999;N.
 30 P.$21;P=A;GOSUB a
 40 P.$6;P=A;GOSUB a
 50 $T="DSIZE";T=T+LEN(T)
 60 ?T=LL0/2560#80;T?1=LL0%256;T?2=#80;T=T+2
 70 A=P;T!1=A
 80 END
 90
100a;[
110:LL0;JSR #E223         ZET CATALOG IN GEHEUEGEN
115     LDA@#02;STA #52   ZET CATALOG RUIMTE IN DE TELLER
120     LDX @#00;STX #53
130     STX #54;STX#55;STX #04
140     LDY #2105;STY #D8 LAADT AANTAL FILES
150:LL1;LDY #D8;BEQ LL2   EINDE INDIEN ALLE FILES VERWERKT
170     JSR #E109         ZET Y-REGISTER OP VOLGENDE FILE
180     STY #D8           SAVE AANTAL FILES
190     CLC
200     LDA @#FF
210     ADC #210C,Y       ZET CARRY VOOR ONVOLLEDIGE SECTOR
220     LDA #210D,Y
230     ADC #52       TEL LAGE BYTEER BIJ OP
240     STA #52       ZET IN TELLER
250     PHP           SAVE DECARRY
260     LDA #210E,Y   LAADT HOGE BYTE AANTAL SECTOREN
270     JSR #E0FB     ELIMINEER LAGE NIBBLE
280     PLP           HAAL DE CARRY TERUG
290     ADC #53       TEL HOGE BYTE ER BIJ OP
300     STA #53       ZET IN TELLER
310     BPL LL1       VOLGENDE FILE
320:LL2;LDY @#52
330     JSR #C99F     ZET WAARDE IN BASIC-STACK
340     JSR #C589     PRINT WAARDE DECIMAAL UIT
341     JMP #C55B     EINDE ROUTINE
350]
360RETURN
```

Dit programma voert een Cyclic Redundancy Check (CRC) uit.
Het doet precies het zelfde als het basic programma op Blz. 93 van Th. & Pr., maar dan niet in 6 minuten maar in een paar seconden. Het is zo geschreven dat het gemakkelijk in een P-CHARME tabel opgenomen kan worden.
Je kunt het gebruiken voor de controle van ingeladen programma's, maar ook als verify van een programma dat net gesaved is.
Dan laadt je dat programma bv. even op #8200 en voert op het origineel en het copy de CHECK uit. Als beide overeenkomen dan staat het goed op de band.
Je kunt er ook geheugen meetesten door een stuk geheugen naar een ander stuk geheugen te copieren en dan op beiden de CHECK uit te voeren.
Tevens kan het gebruikt worden voor de controle van ROM en EPROM. Daartoe is hierbij een overzicht van enkele toolkits gegven.

```
  10 PROGRAM CHECK
  20
  30 DIMJJ4;F.I=0TO4;JJI=999;N.
  40 P.$21;P=A;GOS.a
  50 P.$6 ;P=A;GOS.a
  60 $T="CHECK";T=T+L.T
  70 ?T=JJ0/2560#80;T?1=JJ0%256;T?2=#80;T=T+2
  80 A=P;T!1=A
  90 END
 100
 110a[
 120:JJ0;JSR #F7D1;]
 130$P="BEGIN ADRES : ";P=P+L.P;[;NOP
 140      LDA @CH"#";JSR #CD0F
 150      LDY @#00
 160      LDX @#90;JSR #F893;JSR #F7D1;]
 170$P="EIND ADRES  : ";P=P+L.P;[;NOP
 180      LDA @CH"#";JSR #CD0F
 190      LDY @#00
 200      LDX @#92;JSR #FB93
 210      LDY @#00;STY #A0;STY #A1
 220:JJ1 JSR JJ2
 230      LDX @#90;JSR #FA08;BNE JJ1
 240      JSR JJ2
 250      JSR #F7D1;]
 260$P="SIGNATURE   : #";P=P+L.P;[;NOP
 270      LDX @#A0
 280      JSR #F7F1;JSR #FFED;JMP #C55B
 290:JJ2;LDX @#08
 300      CLC;LDA(#90),Y
 310:JJ3;LSRA
 320      ROL #A0;ROL #A1;BCC JJ4
 330      PHA
 340      LDA #A0;EOR @#2D;STA #A0
 350      PLA
 360:JJ4;DEX;BNE JJ3
 370      RTS
 380];RETURN
```

#C000 - #CFFF	#D67D
#D000 - #DFFF	#AAA1
#F000 - #FFFF	#E386
JOSBOX\AXR-1	#CCEB
WORD-PACK	#83A5
ATOMCALC	#4B9A
SOFTTOOL A&F	#4B66
PP-TOOLKIT	#A3C8
SOFTTOOL I	#1A9F
SOFTTOOL II	#CCC2
WILLOWBOX	#F315
SUPERBASIC	#B1BD
ASBK-I	#43DE
ROMARBOX	#C204
WEBOX-1	#9831

ZUIG-MUIS

De werkgroep "kunsmatige intelligentie" uit Lelystad is eind 1983 begonnen aan een tweede projekt. De aspiraties gingen uit naar het maken van een computer-gestuurd apparaat. Om niet al te hoog te grijpen werd gekozen voor een eenvoudig huishoudelijk apparaat met slechts een afgebakende taak. Dit heeft geresulteerd in het maken van een kleine stofzuiger, "de zuigmuis".
Deze muis heeft slechts een beperkte intelligentie, is eigenlijk heel dom, maar wordt bestuurd door een computer die wel over voldoende programmatuur en geheugen beschikt om de muis zijn taak te kunnen laten volbrengen.
De achterliggende gedachte is dat deze zogenaamde huiscomputer ook tegelijkertijd andere huishoudelijke handelingen kan besturen, zoals het doen van de afwas of het tafeldekken door een robot.

Nu, ongeveer een half jaar later, is het projekt zover gevorderd dat er een mobiel, weliswaar nog niet zuigend karretje gereed is dat draadloos de "zuigmuis" simuleert, alsook de twee computerprogramma's die dit bewerkstelligen.
Aangezien onze interesse zich voornamelijk toespitst op het programmeren en de elektronica, is voor ons het projekt hierbij afgesloten. Een werkelijke realisatie ligt nu verder op het mechanische vlak.

Ter illustratie en verduidelijking van het gekozen projekt wordt in dit artikel een programma beschreven waarin een muis op het t.v.scherm een doolhof zuigt. Dit programma is een vereenvoudiging en samenvoeging van twee programma's voor gebruik op een computer.

De totstandkoming.

Om bovengenoemd resultaat te bereiken is door de leden van de werkgroep aan de volgende onderdelen gewerkt:
- de programmatuur voor beide computers
- de kommunikatie tussen de huiscomputer en de zuigmuis
- een simulatie-model
- het mechaniek van de muis.

- Vooral in de beginfase gebruikten we een simulatie-model. De bewegingen van de zuigmuis konden daarmee zichtbaar worden gemaakt op een t.v.scherm waarop eerst een willekeurig doolhof was opgebouwd.
De kamers die gezogen moeten worden zijn voor de muis immers een onbekende omgeving waarin hij zijn weg moet zien te vinden. De huiscomputer maakt door middel van opdrachten kenbaar welke weg de muis moet volgen.
- In de simulatie-fase werden twee Acorn Atoms gebruikt. Later is een KIM-microprocessorkit met 1 K geheugenuitbreiding in de zuigmuis ingebouwd. Het programma hiervoor is geschreven in machinetaal.
Bij het verdelen van de intelligentie is er van uitgegaan dat de muis bepaalde opdrachten zelfstandig moet kunnen uitvoeren. Deze zijn: Doe stap in richting en Geef informatie over de direkte

omgeving en de eigen konditie.

- De kommunikatie tussen de huiscomputer en de muis werd, in de simulatiefase, simpel verkregen door middel van het kruiselings doorverbinden van de cassette in- en outputs. Met de in COS aanwezige mogelijkheden kon nu data-overdracht plaatsvinden. Een nadeel van deze overdracht is wel dat als de ene computer zendt (=PUT), de andere moet ontvangen (=GET). Met enkele wachtlussen is dit ondervangen. In een latere versie, met de KIM als muiscomputer is dan ook gekozen voor de ovedracht met UART's.

- De muis bestaat uit een frame op twee wielen, aangedreven door stappenmotoren, en een zwenkwieltje (het rolstoelprincipe). Hierdoor wordt een grote wendbaarhed verkregen.

In de muis bevindt zich de KIM-computer, interfaces, een zender en ontvangertje en de akku's. Tevens beschikt de muis over een kompasprogramma dat precies zijn richting bijhoudt. Eventuele storingen worden door de muis doorgegeven aan de huiscomputer.

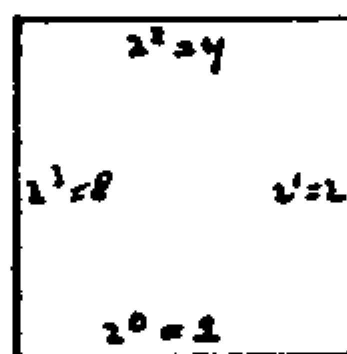
Het programma

Onderstaand programma is geschreven in Acorn Atom Basic. Het is 4 Kbytes lang en vereist 2 Kbytes aan array's, die gelegen zijn in het video-geheugen.

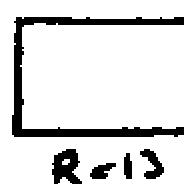
De struktuur van het programma is zodanig dat de afzondelijke programma's van de huiscomputer en de zuigmuis nog herkenbaar zijn. Het muisgedeelte noemen we voor het gemak "de muis", het gedeelte van de huiscomputer "het huis".

De overdracht tussen beide programma's werkt met kommando's van het huis naar de muis en omgekeerd met informatie van de muis naar het huis. Het is de taak van het huis om de zuigmuis de weg aan te wijzen in het doolhof en hem overal te laten komen. Daartoe geeft het huis stapopdrachten in een bepaalde richting aan de muis. Iedere stap komt overeen met de verplaatsing naar een aangrenzend veld van het doolhof. De muis voert deze opdracht uit en geeft daarna de waarde van het array-element van het nieuwe veld en zijn konditie in een statusbyte door aan het huis.

Om te beginnen wordt in de initialisering een random doolhof gegenereerd. Dit doolhof is opgebouwd als een raster waarin op de rasterlijnen al of geen wandjes staan. Ieder veld binnen dit raster wordt in array VV ingedeeld als funktie van zijn x- en y-coordinaten. Het array-element bestaat uit de aanwezigheid van wandjes met een binaire getalswaarde, te weten 2↑0, 2↑1, 2↑2 en 2↑3. In 4 bits kunnen nu alle 16 mogelijkheden van een veld benoemd worden. Het wandje tussen twee naast elkaar liggende velden wordt zowel bij het ene als bij het andere veld meegerekend. Hierdoor wordt vermeden dat de muis bij het geven van informatie over de omgeving ook de aanwezigheid van wandjes in de aangrenzende velden moet nagaan. Hij kan nu volstaan met het geven van de waardevan het array-element van het veld waarin hij zich bevindt.



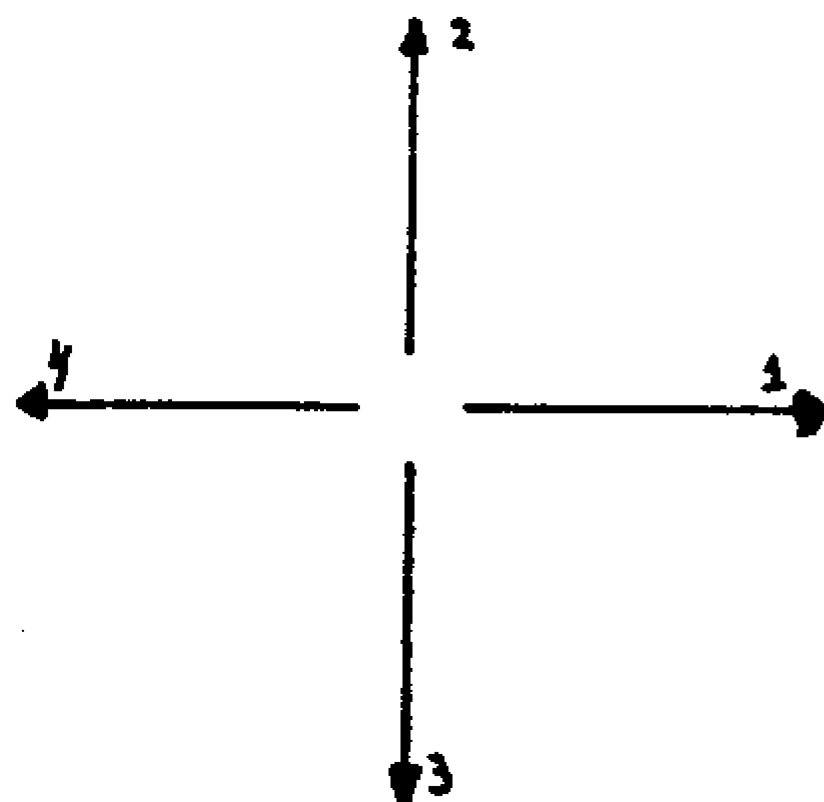




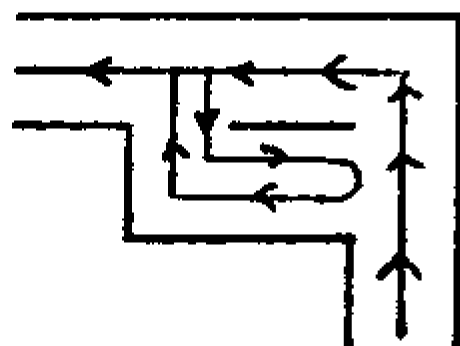


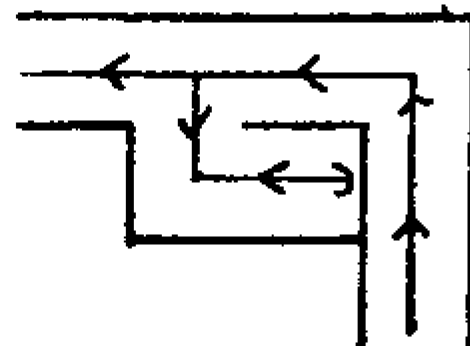
In het agoritme om wegwijs te worden in het doolhof is met het volgende rekening gehouden:

- Om ergens doelbewust te komen, in dit geval overal in het doolhof, moet er een voorkeursrichting worden aangehouden (zie illustratie).



- Bij het zuigen van het doolhof kunnen zich de volgende situaties voordoen: de muis komt op een plaats waar hij al eerder is geweest (deze situatie noemen we een rondloper) of de muis loopt dood (dit noemen we een doodloper).





Beide situaties maken het noodzakelijk dat wordt bijgehouden waar de muis is geweest. Dit gebeurt in array BB met de waarden T (T=afstand tot terminalpunt): BB(x,y)=T. Als T=200 dan is de muis nog niet eerder op dat veld geweest.

- Zowel bij een rondloper als bij een doodloper moet de muis ontsnappen door terug te stappen. De muis zal hierbij zover teruggaan tot hij een toegankelijk veld tegenkomt waar hij nog niet is geweest.
- Het ontsnappen brengt een stapvolgorde met zich mee. Deze bestaat uit array CC met S elementen met daarin de betreffende coordinaten: CC(S)=x,y. In geval van een ontsnapping zal nu alleen S verlaagd hoeven te worden. De reden dat er twee array's zijn die de funktie "teuglopen" hebben komt omdat bij het terugkeren naar het terminalpunt veelal een andere route gevolgd moet worden dan bij een ontsnapping.
- De muis kan door middel van zijn konditie in een statusbyte kenbaar maken dat terugkeer naar het terminalpunt gewenst is. Dit kunt u s(t)imuleren door een toets in te drukken. In het doolhof neemt de muis dan de kortste weg terug naar het terminalpunt, wacht daar even en vervolgt zijn zuigopdracht.
- Om terug te kunnen keren naar het terminalpunt wordt in array BB de volgende truc toegepast. Bij iedere stap kijkt het huis of de muis in de aangrenzende velden is geweest. Heeft een van deze velden een lagere afstandswaarde naar het terminalpunt dan het veld waar hij is, dan wordt de afstandswaarde in array BB aangepast.
- Wanneer de muis overal is geweest zal zijn ontsnaproute de terugkeer naar het terminalpunt zijn. Dit is de aanwijzing dat de boel is aangeveegd.

De gebruikte variabelen zijn:

```
A     - status & omgeving
D     - grootte van doolhof/kamer
I     - momentele waarde aangrenzende BB(I)
J     - flagwaarde wel/geen ontsnappen
M & N - plaats van de muis
Q     - teller
R     - richting + doolhofveldwaarde bij INIT
S     - ontsnapstap-teller
T     - terugstap-teller
U     - coordinaten waarde
V & W - nieuwe veldcoordinaten
X & Y - momentele veldcoordinaten
Z     - konditie
```

De array AA, BB, CC, DD en VV die in de tekst genoemd worden, zijn in het programma teruggebracht tot byte-array's die gepeek't en gepoke't worden.
De array's beginnen op de volgende adressen:

```
?#8C00 - AA() = doolhof volgens het huis
?#8D00 - BB() = terugloop stap-teller BB(U)=T
?#8E00 - CC() = ontsnap route         CC(S)=U
?#8F00 - DD() = terugloop route       DD(per stap)=U
?#9000 - VV() = werkelijke doolhof.
```

```
  10  REM HET DOOLHOF AANGEVEEGD

  20  REM VERSI : ACORN NIEUWS
  30  REM 14 JULI 1984
  35  REM coordinatie S.Verhoef:tel. 03200-49736
  40  LINK#AF00:REM TOOLKIT ROM AAN
  50  REM VOOR COLOUR IS FLP.ROM NODIG
 100 REM INIT
 110  GOS.9100:REM INITIALISERING MUIS
 120  GOS.9200:REM GENEREREN DOOLHOF
 130  GOS.9500:REM UITPRINTEN DOOLHOF
 140  GOS.9000:REM INITIALISERING HUIS
 200 REM hoofdprogramma
 210  GOS.3000:Z=A%16:A=A/16:REM VRAAG STATUS
 215  ?(#8C00+X+15*Y)=A
 220  F=0
 230  GOS.1000:REM BEPAAL RICHTING EN NIEUWE COORDINATEN
 240  GOS.2000:REM MUIS DOET STAP IN RICHTING
 242  IF F=1 GOTO 300
 245 REM UITPLOTTEN DOOLHOF VOLGENS HUIS
 250  E=?(#9000+V+15*W)
 260  MOVE (8*V+4),(8*W+4) ;IF E&1 THEN PLOT 1,8,0
 270  MOVE (8*V+4),(8*W+4) ;IF E&8 THEN PLOT 1,0,8
 280  MOVE(8*V+12),(8*W+4) ;IF E&2 THEN PLOT 1,0,8
 290  MOVE (8*V+4),(8*W+12);IF E&4 THEN PLOT 1,8,0
 300 REM BIJHOUDEN ONTSNAP-COORDINATEN-ARRAY
 310  IF F=0 THEN S=S+1;?(#8E00+S)=V+15*W
 320  IF F=1 THEN S=S-1
 400 REM BIJHOUDEN TERUGLOOP-STAPPEN-TELLER-ARRAY
 410  GOS.3000;Z=A%16;A=A/16:REM VRAAG STATUS
 430  T=T+1
 440  U=V+15*W
 450  IF A&8=0 THEN I=?(#8D00+U-1) ;IF I<T THEN T=I+1
 460  IF A&4=0 THEN I=?(#8D00+U+15);IF I<T THEN T=I+1
 470  IF A&1=0 THEN I=?(#8D00+U-15);IF I<T THEN T=I+1
 480  IF A&2=0 THEN I=?(#8D00+U+1) ;IF I<T THEN T=I+1
 490  ?(#8D00+U)=T
 500 REM BIJHOUDEN NIEUWE PLAATS
 510  X=V;Y=W
 520  ?(#8C00+X+15*Y)=A
 540 REM Z=KONDITIE
 550  IF S=0 LINK#FFE3;RUN
 580  IF Z=0 GOTO 220:REM ZUIG VERDER
 600 REM TERUGLOOP NAAR TERMINAL-PUNT
 610  FOR Q=T-1 TO 0 STEP -1
 620   U=X+15*Y;A=?(#8C00+U);?(#8F00+Q)=U
 622   IF A&8=0 IF ?(#8D00+U-1)=Q ;V=X-1;W=Y   ;R=8
 624   IF A&4=0 IF ?(#8D00+U+15)=Q;V=X   ;W=Y+1;R=4
 626   IF A&1=0 IF ?(#8D00+U-15)=Q;V=X   ;W=Y-1;R=1
 628   IF A&2=0 IF ?(#8D00+U+1)=Q ;V=X+1;W=Y   ;R=2
 640   GOS.2000:REM "GA NAAR"
 650   X=V;Y=W
 655   BEEP 20,6
 660  NEXT
 670 REM TERUG OP TERMINALPUNT
 680  FOR Q=40 TO 10 STEP -1;BEEP Q,2:N.
```

```
 700 REM EN WEER TERUG
 710  FOR Q=1 TO T
 720   U=?(#8F00+Q-1)
 730   V=U%15;W=U/15
 735   GOS.1400
 740   GOS.2000;REM "GA NAAR"
 750   X=V;Y=W
 755   BEEP 10,1
 760  NEXT
 770  A=?(#8C00+X+15*Y)
 790 REM EN MAAR WEER VERDER ZUIGEN
 799 GOTO 220

1000 REM BEPAAL RICHTING
       A=OMGEVING, X EN Y DE PLAATS COORDINATEN
1030  U=X+15*Y;R=0
1040  IF A&8=0 IF ?(#8D00+U-1)=D ;V=X-1;W=Y  ;R=8
1050  IF A&4=0 IF ?(#8D00+U+15)=D;V=X  ;W=Y+1;R=4
1060  IF A&1=0 IF ?(#8D00+U-15)=D;V=X  ;W=Y-1;R=1
1070  IF A&2=0 IF ?(#8D00+U+1)=D ;V=X+1;W=Y  ;R=2
1080  REM IN R DE VOORKEURSRICHTING
1090  IF R=0 THEN U=?(#8E00+S-1);V=U%15;W=U/15;GOS.1400;F=1
1099 RETURN

1400 REM MAAK RICHTING UIT NIEUWE COORDINATEN
1420  IF V<X THEN R=8
1430  IF V>X THEN R=2
1440  IF W<Y THEN R=1
1450  IF W>Y THEN R=4
1499 RETURN

2000 REM MUIS DOET STAP
2010  PLOT 15,(M*8+8),(N*8+8);REM HAAL MUIS VAN HET SCHERM
2020 REM BEPAAL COORDINATEN NIEUWE PLAATS
2030  IF R=1 THEN N=N-1
2040  IF R=2 THEN M=M+1
2050  IF R=4 THEN N=N+1
2060  IF R=8 THEN M=M-1
2070  IF F=1 THEN BEEP 30,1
2080 REM ZET MUIS WEER OP HET SCHERM
2090  PLOT 14,(M*8+8),(N*8+8)
2100  PLOT 13,(M*8+6),(N*8+6)
2999 RETURN

3000 REM MUIS GEEFT STATUS
3010 REM KONDITIE = (BIT 0 T/M 3 )
3020  KEY A;IF A<>0 THEN A=1
3030 REM OMGEVING = (BIT 4 T/M 7 )
3090  A=?(#9000+M+15*N)*16+A
3099 RETURN
9000 REM INITIALISERING HUIS
9010  D=200
9020  S=0;T=0;X=0;Y=0;V=0;W=0
9030 REM SCHOONMAKEN BYTE-ARRAY'S
9040  FOR I=0 TO D;?(#8D00+I)=D;?(#8C00+I)=D;?(#8F00+I)=0;N.
9050  ?(#8D00+X+15*Y)=0;?(#8E00+0)=0
```

```
9090 RETURN

9100 REM INITIALISERING MUIS
9110  K=0:L=0:M=0:N=0
9190 RETURN

9200 REM GENERERING DOOLHOF
9210  P.$12'''"IK BOUW DOOLHOF"'''"EVEN GEDULD S.V.P."
9220  P.''
9230  FOR X=0 TO 14
9240   FOR Y=0 TO 10
9250    R=ABS(RND%15)
9260    IF X=0 THEN R=8
9270    IF X=14 THEN R=2
9280    IF Y=0 THEN R=RØ1
9290    IF Y=10 THEN R=RØ4
9300    IF X=13 THEN R=R&13
9310   REM NALOPEN AANGRENZENDE VELDEN
9320    IF X=0 GOTO 9350
9330    IF ?(#9000+(X-1)+Y*15)&2=0 THEN R=R&7
9340    IF ?(#9000+(X-1)+Y*15)&2=2 THEN R=RØ8
9350    IF Y=0 GOTO 9400
9360    IF ?(#9000+X+15*(Y-1))&4=4 THEN R=RØ1
9370    IF ?(#9000+X+15*(Y-1))&4=0 THEN R=R&14
9380    IF Y=10 GOTO 9410
9390    IF X=14 GOTO 9410
9400    IF R=15 OR R=14 OR R=13 OR R=7 OR R=11 GOTO 9250
9410    IF R=15 AND X<14 GOTO 9250
9420    ?(#9000+X+15*Y)=R
9430    ?(#9000+0)=11
9440   NEXT
9450   P.14-X,$13
9460  NEXT
9490 RETURN

9500 REM UITPLOTTEN VAN DOOLHOF
9510  CLEAR 3:COLOUR 3
9520  FOR X=0 TO 14
9530  FOR Y=0 TO 10
9540  R=?(#9000+X+15*Y)
9550  MOVE (8*X+4),(8*Y+4) :IF R&1 THEN PLOT 1,8,0
9560  MOVE (8*X+4),(8*Y+4) :IF R&8 THEN PLOT 1,0,8
9570  MOVE(8*X+12),(8*Y+4) :IF R&2 THEN PLOT 1,0,8
9580  MOVE (8*X+4),(8*Y+12):IF R&4 THEN PLOT 1,8,0
9590  NEXT:NEXT
9595  COLOUR 1
9599 RETURN
```

Onderstaand programma maakt het mogelijk om procedures en functies (zie beschrijving P-CHARME box) te koppelen aan een ander programma. Je zou nu een hele procedure- en functie-bibliotheek op kunnen bouwen waar je de benodigde procedures en functies uitlicht en koppelt aan het programma waarin je deze nodig hebt. De plaats waar het "hoofdprogramma", het "koppelprogramma" en de "procedure- en functie-bibliotheek" in het geheugen staan zijn vrij te kiezen. Wanneer het woord "KOPPEL" wordt ingetypt vraagt de Atom in welk geheugengebied hij moet gaan zoeken naar procedures en functies. Wanneer dit opgegeven is begint de zoekaktie en wordt een genummerde lijst gegeven van alle in het opgegeven geheugengebied gevonden procedures en functies. Vervolgens wordt gevraagd vanaf welk adres gezocht moet worden naar het einde van het "hoofdprogramma". Wanneer dit gevonden is wordt gevraagd naar de nummers van de procedures en functies die gekoppeld moeten worden. Als alle gewenste procedures en functies zo gekoppeld zijn wordt als nummer 0 opgegeven. Spring nu naar het hoofdprogramma en geef een RENUMBER commando. Zet voor de eerste gekoppelde procedure of functie een label, bijv. q. Ergens aan het begin van het programma moet dan nog GOS.q worden gezet en de zaak is rond. Dit programma werkt gegarandeerd plezieriger dan het telkens weer intypen van je standaard functies en procedures of dan met de koppelmethode van Atomic Theory & Practice hoofdstuk 19.5. Een heel elegante alternatieve methode wordt beschreven in A.N. mei'84, blz.48 en 49 m.b.v. de statements ATTACH en EXTERN. De moeilijkheid hierbij is alleen dat altijd samen met het "hoofdprogramma" ook de "bibliotheek" moet worden geladen, voordat het hoofdprogramma kan worden ge-"run"-d. Clubgenoten die jouw programma willen gebruiken moeten dan ook de beschikking hebben over (een deel van) jouw bibliotheek. Vandaar het programma KOPPEL.

```
   10 PROGRAM KOPPEL
   20 REM HANS HEGT, TEL.04192-16239
   30
   40 PROC ZOEK,a
   50  a=2;P=A
   60b D=INSTR($P,$C)
   70  IF D>0;GOTOc
   80  P=P+LENP+1;IF P<B;GOTOb
   90  GOTOf
  100c PP(N)=P;P.N"  "$(P+2)'
  110g D=INSTR($P,$E);P=P+LENP+1
  120  IF D>0 GOTOe
  130  IF P<B GOTOg
  140  P.$E" NIET GEVONDEN!"';GOTOf
  150e QQ(N)=P;N=N+1;P=P+LENP+1;GOTOb
  160fPEND
  170 N=1;DIM C(4),E(4),PP(20),QQ(20)
  180 IN."ZOEKEN VANAF ADRES"A
  190 IN."TOT ADRES"B
  200 $C="PROC";$E="PEND";ZOEK
  210 $C="FUNC";$E="FEND";ZOEK
  220 IN."VANAF WELK ADRES ZOEKEN NAAR"'"PROGRAMMATOP"P
  230 DO;P=P+LENP+1;T=P;UNTIL?P=#FF
  240hREM T=ADRES VAN BYTE #FF
  250 IN."WELKE FUNKTIE OF PROCEDURE"'"KOPPELEN"A
  260 IF A=0;$(T-1)=";R.";?(T+3)=#FF;END
  270 COPY PP(A),QQ(A),T;T=T+QQ(A)-PP(A)
  280 P."GEKOPPELD:"'"  "$(PP(A)+2)';GOTOh
```

Alvorens met de programma's te kunnen werken zal met behulp van FCAT-1 een dummyfile (FCAT) moeten worden aangemaakt.
FCAT-1 maakt tevens het ramgeheugen gereed voor gebruik door het programma SETCAT.
Het programma SETCAT neemt van alle door u gewenste discs de catalog over in een bestand.
Nadat het door SETCAT aangemaakte file gereed is en door het programma is gesaved onder de naam FCAT kan het door DISCAT worden gepresenteerd op het scherm.
Met een keuze mogelijkheid om de catalogi van alle resp. een enkele schijf te lezen of een bepaald programma in het bestand te zoeken en aan te geven op welke schijf dit programma zich bevind.Het is gebleken dat het uiterst handig is de schijven te nummeren.

```
 10REM FCAT-1
 20REM VERSIE 1                          FRANS LE BLANC
 30P.$12"WANNEER ER VOOR DE EERSTE KEER  EEN FCAT GEMAAKT WORDT"
 40P." MOET DIT PROGRAMMA GERUND WORDEN"''
 50P."DOE DE DISK IN DE DRIVE"'"DRUK OP EEN TOETS"'
 60LINK#FFE3
 70FILL#3000,#67FF,0
 80F.N=#3000TO#67FFS.#108
 90!N=#20202020;N!4=#20202020
100!(N+#100)=#20202020;N.
110LINK#E000
120*S.FCAT 3000 67FF
130E.
```

```
 10REM SETCAT
 20REM VERSIE 1                          FRANS LE BLANC
 30LINK#E000;*L.FCAT
 40DIMB1
 50DO P.$12"INSERT DISK"'
 60VTAB6;P."PRESS CONTROL FOR STOP";CU.0,2
 70IN."TYPE DISKNUMBER"$B
 80IFVALB<=0 OR $B="" OR L.B>2;P.$11;LINK#FE22;G.70
 90A=VALB
100IFA>54;P."MAX.54 DISK'S!"'$11$11;LINK#FE22;G.70
110LINK#E223;Q=?#2105+#2008
120FILL Q,#20FF,0
130B=A*#108+#2EF8
140COPY#2000,#2107,B
150U.?#B001=191;P.$12"INSERT DISK FOR SAVING FCAT"';LINK#FFE3
160*N.
170*S.FCAT 3000 67FF
180P."FCAT SAVED"''"READY"';E.
```

In het vorige nummer is reeds het programma DISKAT gepubliceerd, zonder dat de redaktie in de gaten had dat bovenstaande programma's er bij hoorden. exuses hiervoor.

ELEKTUUR-MODEM

Als eerste blad in Nederland publiceerde Elektuur in het Septembernummer een direct gekoppeld zelfbouw telefoonmodem. Dit modem werkt volgens de V21 en V23 normen.

Omdat ikzelf bij de ontwikkeling betrokken ben geweest is het voor mij gemakkelijk enige eigenschappen van dit modem toe te lichten. Het zal voor iedereen duidelijk zijn dat er geen reden is om nog zelf een modem te maken volgens een eigen ontwerp. De mogelijkheden van dit modem zijn zodanig dat iedereen hiermee tevreden zal zijn.

De mogelijkheden;

De ingang van het modem is RS 232 kompatible, verder is er een ingang op TTL niveau zodat men ook zonder de RS 232 spanningsniveau's kan werken.
Dit betekend voor de ATOM dat met een 6551 (ACIA) en een 7404 op plug 8 een modemaansluiting te maken is (dus zonder een ekstra voeding), zie hiervoor het bijgesloten schema. Bij dit schema is er vanuit gegaan dat ook Big-Benny op plug 8 blijft, er is nu nog plaats voor twee pia's of andere interfaces die vier geheugen plaatsen nodig hebben. Het modem kent vier werkstanden namelijk twee standen van 300 baud full duplex en twee standen in Viditelmode met een maincannel van 1200 baud en een backcannel van 75 baud.
Men kan nu dus met 1200 baud een programma overzenden, terwijl de ontvanger met 75 baud kommentaar terug zendt. In deze mode zendt de computer met 1200 baud terwijl de interspeeder de omzetting naar 75 baud verzorgt. De gemiddelde snelheid voor het backcannel is echter wel 75 baud, terwijl de data met 1200 baud aangeboden kan worden. Hiertoe is op de TTL plug een handshake signaal aanwezig in de vorm van de TMBT pen van de UART.
Deze interspeeder spaart een extra klok voor de ACIA uit.

Behalve de stand modem is er ook nog eem stand autoanswer, in deze stand is een automatisch dataverkeer mogelijk, want als nu iemand het modem opbelt, wordt er opgenomen en meldt het modem zich met een carrier welke binnen een bepaalde tijd beantwoord moet worden. Gebeurt dit dan is er een dataverbinding gebeurt dit niet dan wordt door het modem de lijn weer vrij gemaakt, dit gebeurt meestal binnen zo'n 10 seconden.

Op deze manier is het dus mogelijk binnen de gebruikersgroep een databank te maken waaruit iedereen data kan halen, maar dan wel volledig automatisch.

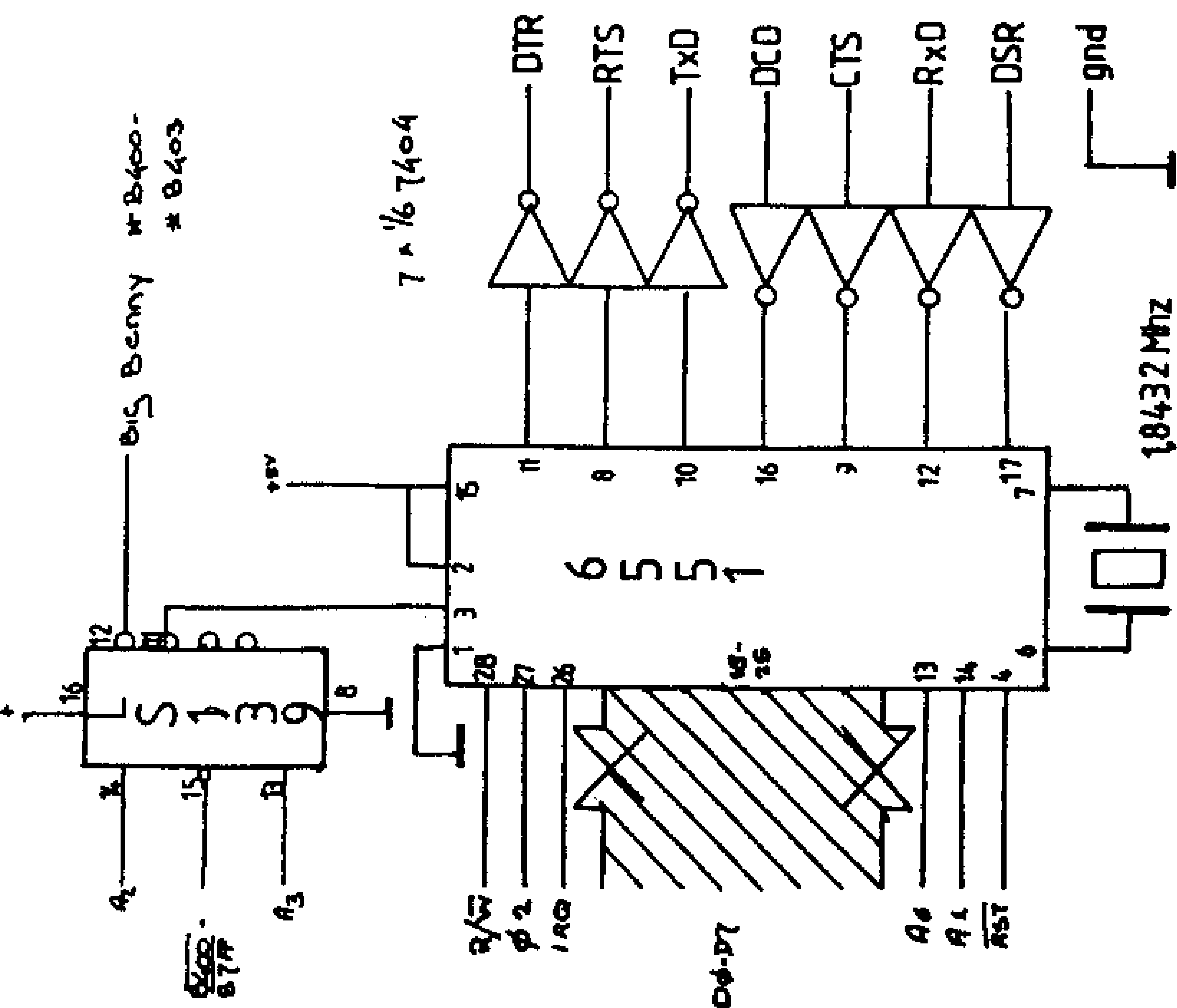
Het modem is zodanig gekonstrueerd dat het onder een telefoon kan staan.

Voor de gebruikersgroep is nu de dag aangebroken om eens aan de slag te gaan en nu een software pakket te maken zodat er tussen de gebruikers programma's uitgewisseld kunnen worden en dat de Atom

als terminal bruikbaar is. Vanaf nu is de Atom dus geschikt om uit de verschillende databanken hier in Nederland en ook in Duitsland data op te halen. Het gebruik als Viditel terminal is niet direkt mogelijk vanwege de speciale karakterset die hiervoor nodig is, er bestaat echter voor de Atom een Viditel VDU kaart die voortgekomen is uit het system 1 projekt, de software voor deze kaart zit in de DOS en het schema van de kaart alsmede de print lay-out is binnen de gebruikersgroep beschikbaar. Het ontvangen van Viditel is geen probleem alleen het weergeven van de speciale karakters van viditel is moeilijk voor de Atom.

Voor de toekomst is de uitbreiding van het modem met een autodailer zeer zeker zinvol alsmede het aanbrengen van current loops om een Telex aan te sluiten of een verbinding over grotere afstand, tot max. ca. 1 Km.
Er moet vanuitgegaan worden dat pulsdailers de einge dailers zijn die voor iedereen toepasbaar zijn.

Ik hoop van harte dat wij met dit modem veel computergebruikers een goed stuk gereedschap hebben geleverd en wens iedereen veel plezier toe met het ELEKTUUR VOLKSMODEM. Voor op- of aanmerkingen houd ik mij gaarne aanbevolen.



MCAT geeft de diskcatalog in drie kolommen op het scherm en geeft als aanvullende informatie het aantal gebruikte Kbytes en aantal programma's op de schijf.

```
 10 PROGRAM MCAT
 20 REM  F.G.LE BLANC-MONSTER
 30 REM     01749-42635
 40 REM ****************************
 50 REM * GEEFT 3-KOLOMS DISK-     *
 60 REM * CATALOGUS (*CAT)         *
 70 REM * MET EXTRA INFORMATIE     *
 80 REM ****************************
 90
100 DIMLL17,PP7,SS1
110 F.N=0TO27;LLN=Q;N.
120 P.$21;P=A;GOSUB a
130 P.$6;P=A;GOSUB a
140 $T="MCAT";T=T+L.T
150 ?T=SS0/256■#80;T?1=SS0%256;T?2=#80;T=T+2
160 A=P;P!1=A
170 END
180
190a;[;:SS0
200JSR#E000;JSR#E223;LDA@#C;JSR#FFF4
210LDX@#00;STX#B6
220:LL0;LDA#2000,X;CPX@#08;BCCLL1;LDA#20F8,X
230:LL1;JSR#FFF4;INX;CPX@#0D;BNELL0;JSR#E016
240];$P=" DRIVE ";P=P+L.P;[
250LDA#EE;JSR#F80B
260JSR#E016
270];$P=" QUAL ";P=P+L.P;[
280LDA#AC;JSR#FFF4
290:LL2;LDY@#00;JSRLL5;BCCLL12
300:LL3;JSR#E109;LDA#2008,Y;AND@#7F;STA#2008,Y;TYA
310BNELL3;JSR#FFED;JSR#FFED;JMP PP0
320:LL4;JSR#E100
330:LL5;CPY#2105;BCSLL6;LDA#2008,Y;BMILL4
340:LL6;RTS
350:LL7;LDY#B8
360CPY@2
370BNELL8;JSR#FFED;LDY@#FF
380:LL8;INY;STY#B8;JSRLL17
390:LL9;LDA@#23;LDY#B7;LDX#200F,Y;BMILL10;LDA@#20
400:LL10;JSR#FFF4;LDX@#00
410:LL11;LDA#AE,X;JSR#FFF4;INX;CPX@#07;BNELL11
420BEQLL2
430:LL12;STY#B7;LDX@#00
440:LL13;LDA#2008,Y;AND@#7F;STA#AE,X;INY;INX
450CPX@#08;BNELL13
460:LL14;JSRLL5;BCSLL16;LDX@#06;SEC
470:LL15;LDA#200E,Y;SBC#AE,X;DEY;DEX;BPLLL15
480JSR#E101;LDA#200F,Y;AND@#7F;SBC#B5;BCCLL12
490JSR#E100;BCSLL14
500:LL16;LDY#B7;LDA#2008,Y;ORA@#80;STA#2008,Y;LDA#B5
510CMP#B6;BEQLL7;STA#B6;JSR#FFED;LDA#B5
520JSR#FFF4;LDA@#3A;JSR#FFF4;LDY@0
```

```
530STY#B8;BEQLL9;LDA#210E,Y;JSR#E0FB;STA#A2
540CLC;LDA@#FF;ADC#210C,Y;LDA#210F,Y;ADC#210D,Y
550STA#A3;LDA#210E,Y;AND@#0F;ADC#A2;STA#A2
560SEC;LDA#2107,Y;SBC#A3;PHA;LDA#2106,Y
570AND@#0F;SBC#A2;TAX;LDA@#00;CMP#A0
580PLA;SBC#A1;TXA;SBC@#00;RTS
590:LL17 LDY@2;JSR#E0EC;DEY;BNE LL17+2;RTS
600:PP0 LDA#321;PHA;LDA@2;STA#321
610LDY@0;STY#16;STY#25;STY#34;STY#43;JSR#E109
620:PP1 JSR#E100;CPY#2105;BEQPP4
630LDA#210C,Y;CLC;ADC#34;STA#34;BCCPP2;INC#16
640:PP2 LDA#210D,Y;CLC;ADC#16;STA#16;BCCPP3;INC#25
650:PP3 JMPPP1
660:PP4 LDX@0;STX#34;CLC
670ROR#25;ROR#16;CLC;ROR#25;ROR#16;LDA#16;STA#B8
680JSR#F7D1;]
690$P="TOTAL MEMORY: ";P=P+L.P;[;NOP
700JSR#C589;JSR#F7D1;]
710$P=" KBYTES ";P=P+L.P;[;NOP
720LDA#B8;CMP@96;BCCPP5;JSRPP7
730:PP5 JSR#FFED;LDA#2105
740LSRA;LSRA;LSRA;STA#16;STA#B7;JSR#F7D1;]
750$P="TOTAL PROGR.: ";P=P+L.P;[;NOP
760JSR#C589;LDA#B7;CMP@31;BCCPP6;LDY@8;JSR#E0F3;JSRPP7
770:PP6 PLA;STA#321;JSR#FFED;JMP#C558
780:PP7 JSR#F7D1;]
790$P="full";P=P+L.P;[;NOP;RTS
800]
810 RETURN
```

```
 10 PROGRAM RAADSEL
 20
 30 S=#802B
 40 PRINT $12$15
 50 ?#E1=0 ;REM CURSOR UIT
 60
 70 DO
 80  PRINT $S'
 90  N=0
100  BEEP 5,5
110 UNTIL 0
120
130 WAAR KOMEN AL DIE PLUSJES VANDAAN ?
140 HOE KOMT HET DAT DE TEKENS (IN DIT GEVAL DUS PLUSJES)
150 VERBAND HOUDEN MET HET LAGE BYTE VAN S IN REGEL 50 ?
160 (HIER : CH"+"=#2B = S&#00FF)
170
180 S=#8023 IS OOK EEN LEUKE WAARDE
190
200 OPLOSSINGEN AAN :
210  ROB VAN DORT
220  ELIASSTRAAT 27
230  ALVERNA
240
250 OF NOG BETER: AAN DE REDAKTIE VAN ONS LIJFBLAD
```

De fastcos routines van de TOOLKIT, JOSBOX en P-CHARME blijken nogal kritisch te zijn bij het laden van programma's op 1200 baud. Hoe dit komt en wat er tegen gedaan kan worden is het onderwerp van dit artikel.

Bij het laden van een programma draait alles om e e n routine, nl. de routine die een byte van cassette kan lezen. Door deze routine bij herhaling aan te roepen wordt uiteindelijk het gehele programma geladen. Nu bestaat het lezen van een byte op zijn beurt weer uit het achtereenvolgens lezen van een aantal bits, en hierom gaat het. Wat gebeurt er precies bij het lezen van zo'n bit? Om dit duidelijk te maken ga ik een gedachtenexperiment met u uitvoeren.

Stel u bevindt zich in een grote ronde kamer waarin het absoluut donker is. De vloer van deze kamer draait met een konstante snelheid rond. U staat op deze vloer, vlakbij de wand met uw gezicht naar de wand toe gekeerd. Ergens in de wand bevindt zich op ooghoogte een klein gaatje, waarachter zich een lichtbron bevindt. Door deze lichtbron volgens een bepaalde kode afwisselend rood en groen licht te laten schijnen, worden er achtereenvolgens informatie bits uitgezonden.

We hebben de volgende kode afgesproken:

Een 0 bit wordt voorgesteld door de lichtbron gedurende 1 sekonde 8 maal van kleur te laten veranderen, d.w.z. 1/8 sekonde rood, 1/8 sekonde groen, 1/8 sekonde rood etc.
Een 1 bit wordt voorgesteld door de lichtbron niet 8 maar 16 maal van kleur te laten veranderen in dezelfde tijd (1 sek.).
Merk op dat ieder bit even lang duurt, nl. precies een sekonde. De snelheid is dus 1 bit/sec (1 baud).

We gebruiken de ronddraaiende beweging van de vloer waarop we nog steeds staan als "tijdmeter" om het aantal kleurwisselingen per bit-tijd te bepalen. Hiervoor is het wel nodig dat de vloer snel genoeg ronddraait, b.v. 100 omwentelingen per sekonde (bittijd). Gaat het u nog niet duizelen? Als we gedurende 100 omwentelingen bijhouden hoe vaak de lichtbron van kleur verandert, dan weten we of er een "0" of een "1" wordt uitgezonden. Tellen we nl. 8 kleurwisselingen dan is er sprake van een "0" bit. Voor een "1" bit zullen we er 16 tellen. Als regel houden we aan dat minder dan 12 kleurveranderingen een "0" aangeven, en 12 of meer een "1".
Op deze manier werkt de standaard ATOM COS (op 300 baud).

U heeft zich misschien afgevraagd waarom we de lichtbron zo vaak van kleur laten veranderen om een enkel bit uit te zenden. Kan het niet wat minder? Inderdaad, met 2 resp. 4 wisselingen voor een "0" resp. "1" bit kan het ook. De bittijd wordt dan 4 maal zo kort en dus de snelheid 4 maal zo groot. We gaan weer precies zo te werk als voorheen. Alleen tellen we nu het aantal kleurwisselingen gedurende 25 omwentelingen. De bittijd is immers 4 maal zo klein geworden. Voor een "0" bit zullen we 2 kleurveranderingen tellen en voor een "1" bit 4. We nemen het getal 3 als breekpunt tussen een "0" en "1" bit: voor 2 of minder wisselingen noteren we een "0". Voor 3 of meer noteren we een "1".
Op deze manier werken zowel de TOOLKIT, JOSBOX als de P-CHARME op 1200 baud.

Wat is nu het probleem? Stel dat onze lichtbron af en toe te vroeg of te laat van kleur wisselt. We kunnen dan gemakkelijk een kleurverandering te weinig of te veel tellen, waardoor er leesfouten zullen ontstaan. Ga dit zelf maar na en bedenk hierbij dat er net een kleurwisseling kan hebben plaatsgevonden op het moment dat we met onze telling beginnen, of dat we meteen al een verandering zien.

De oplossing van dit probleem is erg eenvoudig. We tellen niet het aantal kleurwisselingen gedurende een bittijd, maar het aantal omwentelingen van de vloer tussen twee kleurwisselingen, d.w.z. een volledige periode rood en groen. Zitten we in een "0" bit dan zullen we ca. 25 omwentelingen tellen en in een "1" bit ca. 12 (25/2) omwentelingen. Tellen we een omwenteling te veel of te weinig dan is dit geen probleem, omdat we voldoende speling hebben. Een "0" bit hebben we meteen al te pakken. Voor een "1" bit moeten we nog 2 kleurwisselingen wachten voordat we het volgende bit kunnen gaan lezen. Op deze manier blijven we ieder bit "op de voet volgen". Ons systeem is zeer ongevoelig voor storingen. In theorie mag een storing zelfs bijna + of - 50% zijn!

Als we ons model "terugvertalen" naar de werkelijkheid, dan komt een kleurverandering overeen met een signaalwisseling op de band. Het aantal signaalwisselingen per sekonde is beperkt door de rekorder en tape eigenschappen. Vandaar dat bij de overgang van 300 naar 1200 baud is gekozen voor dezelfde signaal-frequentie. De vloeromwentelingssnelheid komt overeen met het aantal malen per tijdseenheid dat de Acorn Atom naar de signalen van de rekorder kan kijken. Deze snelheid kan van nature hoger zijn dan de "kleurwisselingssnelheid". Het is daarom mogelijk om redelijk nauwkeurig de tijdsduur tussen twee signaalwisselingen te meten.

Tenslotte volgen de voorgestelde wijzigingen voor de diverse ROM's. Merk op dat voor MJCOS geen wijzigingen nodig zijn, aangezien deze reeds volgens bovenbeschreven methode werkt.
De wijzigingen zijn in principe gelijk voor alle drie de ROM's. Alleen de adressen verschillen. De nieuwe code is als volgt:

```
18          CLC
08          PHP
20 BD FC    JSR #FCBD
20 BF FC    JSR #FCBF
E0 12       CPX @#12
B0 06       BCS P+8
28          PLP
B0 04       BCS P+6
38          SEC
B0 EF       BCS P-15
28          PLP
EA          NOP
EA*         NOP     *** alleen voor TOOLKIT en JOSBOX ***
EA*         NOP     ***     idem
EA*         NOP     ***     idem
```

De beginadressen voor de wijzigingen zijn:

```
TOOLKIT:  #ADD3
JOSBOX :  #A53D
P-CHARME: #ADF3
```

MEER GEHEUGEN

IK ZELF HEB EEN 64K KAART EN HET LEEK ME WEL HANDIG OM GEHEUGEN ONDER DE #1000 TE HEBBEN EN AANGEZIEN IK TOCH 2114'S OVER HAD..... IK HEB EEN PAAR WEKEN MET HET IDEE ROND GELOPEN TOTDAT IK EEN OPLOSSING VOND.
DE IC-VOETJES VAN HET LAGER GEHEUGEN EN VAN IC-6 (DE DECODER HIERVOOR 74LS138) WAREN LEEG. NOU DACHT IK, DIE KAN IK MOOI GEBRUIKEN. IN MIJN DRIVE HAD IK NOG EEN ONGEBRUIKTE 74LS139 (WANT DAAR HAD IK OOK AL GEEN GEHEUGEN IC'S MEER VANWEGE DE 64K KAART) DUS HET LEEK ME WEL VOOR DE HAND LIGGEN OM DIE TE GEBRUIKEN.
DE 74LS139 IS EEN 4 UIT 2 DECODER. DAT HOUDT IN, DAT JE ER EEN 2 BITS BINAIR GETAL INSTOPT EN EEN VAN DE 4 UITGANGEN WORDEN GESELECTEERD. IS HET BINAIRE GETAL 00 DAN WORDT DE EERSTE UITGANG GESELECTEERD;IS HET 01 DE TWEEDE UITGANG ENZ.
DE '139 HEEFT OOK EEN SELECT INGANG. PAS ALS DEZE INGANG LAAG WORDT DAN PAS GAAT DE '139 DECODEREN.
DE GEHEUGEN IC'S IN DE COMPUTERS WORDEN GE-ENABLED (GE-ACTIVEERD) DOOR HET LAAG MAKEN VAN DE CS (CHIP SELECT) LIJN (PEN 8 VAN DE 2114)
WE VOEGEN 2K GEHEUGEN TOE AAN DE COMPUTER , EN EEN 2114 LEVERT 1K*4 BITS. DAT BETEKENT DAT WE 2 2114'S NODIG HEBBEN VOOR 1K*8 BITS EN DUS 4 VOOR 2K GEHEUGEN. DUS VOOR DEZE 2K HEBBEN WE 2 CS SIGNALEN NODIG. DEZE HAAL IK VAN DE '139 WAAR IK HET NET HEB OVER GEHAD.
DE '139 MAG ALLEEN MAAR GAAN DECODEREN ALS DE MICRO-PROCESSOR ONDER DE #1000 ADRESSEERT. DIT KAN DOOR HET SELECT PENNETJE VAN DE '139 AAN TE SLUITEN MET HET BLOCK0 SIGNAAL UIT DE COMPUTER. ALS WE NU A10 EN A11 OP DE TWEE SELECT INGANGEN AANSLUITEN DAN KAN DE '139 4 GEHEUGEN GEBIEDEN ONDER DE #1000 SELECTEREN,NL.:

#0000 TOT #03FF
#0400 TOT #07FF
#0800 TOT #0BFF
#0C00 TOT #0FFF

NOU VAN #000 TOT #3FF IS OA. ZERO-PAGE RAM EN IS AL BEZET. VAN #0A00 T/M #0A03 ZIT DE FDC VAN DE DISKDRIVE DUS DAAR KUNNEN WE OOK GEEN RAM NEER ZETTEN. DUS ZIJN #400 TOT #7FF EN #C00 TOT #FFF ONBENUT DOOR DE COMPUTER EN DAAR ZETTEN WE DAN MOOI RAM OP.

NU VOLGEN ER TWEE BOUW BESCHRIJVINGEN. EEN VOOR MENSEN MET EEN 64K KAART EN EEN VOOR MENSEN ZONDER EEN DERGELIJKE KAART.
VOOR DE 64K KAART MENSEN:

HET EERSTE WAT MEN DOET IS HET PRINTPLAATJE NR.1 MAKEN.(NR.2 IS VOOR MENSEN DIE GEEN 64K KAART HEBBEN). DEZE IS NIET OP WARE GROOTTE GEDRUKT!!!
OP DE PLAATS WAAR XAZ STAAT DAAR KOMT EEN 16 PENS WIREWRAP VOET (ZO EEN IC VOET MET VAN DIE LANGE POTEN). DE POTEN NIET AFKNIPPEN MAAR LANG LATEN WANT DEZE POOTJES KOMEN IN DE IC VOET VAN IC-6.
OP DE ANDERE IC PLAATS KOMT EEN GEWOON IC VOETJE MET DE 74LS139 MET DE INHAM BIJ DE 'G'. SOLDEER AAN 'G' ALVAST EEN DRAADJE VAN 5 TOT 7 CM.
NU STEKEN WE HET PRINTPLAATJE IN HET VOETJE VAN IC-6 ZODAT IC-7 DOOR HET PRINTJE BEDEKT WORDT EN DE LS139 NAAR DE ZELFDE RICHTING WIJST ALS DE NAAST LIGGENDE IC'S.
IN DE VOETJES VAN IC-10,IC-11,IC-16 EN IC-17 DRUK JE NU 4 2114'S. SOLDEER HET DRAADJE VAN PUNT 'G' NU AAN BLOCK0 ENABLE (PEN-3 VAN IC-8 OF PEN-31 VAN PL6/7).

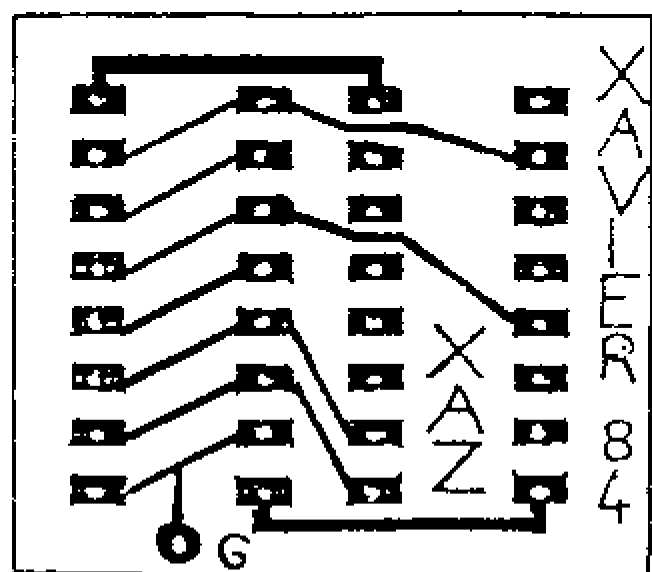
ALS JE DIT ALLEMAAL GOED HEBT GEDAAN MOET JE NU 2K EXTRA RAM HEBBEN ONDER DE #1000.

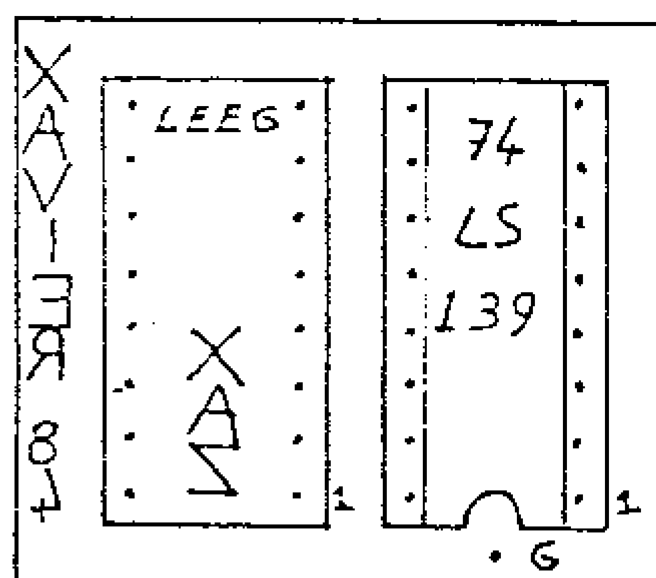
VOOR MENSEN DIE GEEN 64K KAART HEBBEN:
BEGIN MET HET MAKEN VAN PRINTPLAATJE NR.2 (NR.1 IS VOOR MENSEN DIE EEN 64K KAART HEBBEN).
OP DE PLAATS WAAR 'XAZ' STAAT ZET MEN EEN 16 PENS WIREWRAP VOET (ZO EEN IC VOET MET VAN DIE LANGE POTEN). DE POTEN LANG LATEN!!!
DAARNAAST ZET MEN EEN GEWONE 16 PENS IC VOET MET DE 74LS139 ERIN.
IN DE WIRE WRAP VOET ZET MEN DE 74LS138 DIE OP DE PLAATS VAN IC-6 ZIT EN NIET DIE VAN IC-23 OF IC-30!!
ALLEBIJ DE IC'S HEBBEN DE INKEPING AAN DE ZELFDE KANT NL. RICHTING DE 'G' (ZIE TEKENING).
SOLDEER ALVAST DE DRADEN AAN '4','C' EN 'G' (ONGEVEER 5 TOT 7 CM).
NU ZET JE HET PRINTJE OP DE PLAATS VAN IC-6 ZODAT IC-7 BEDEKT WORDT DOOR HET PRINTJE EN DE IC'S NAAR DE ZELFDE RICTING WIJZEN ALS DE NAAST LIGGENDE IC'S (IN DE ATOM).
SOLDEER DE 'G' DRAAD AAN DE BLOCK0 ENABLE (PEN-8 VAN IC-8 OF PEN-31 VAN PL6/7).
NU MOET JE 4 2114'S STAPELEN OP HET LAGER GEHEUGEN EN DRAAD '4' EN 'C' AANSLUITEN MET DE CHIPSELECT LIJNEN (PEN-8) ZIE TEKENING.
ALS JE DIT ALLEMAAL GOED HEBT GEDAAN DAN HEB JE NU 2K ERBIJ.

OPMERKING:
DE 74LS139 BEVAT 2 4 UIT 2 DECODERS. IK HEB ZE ALLEBIJ PARRALEL GESCHAKELD VOOR HET GEMAK.

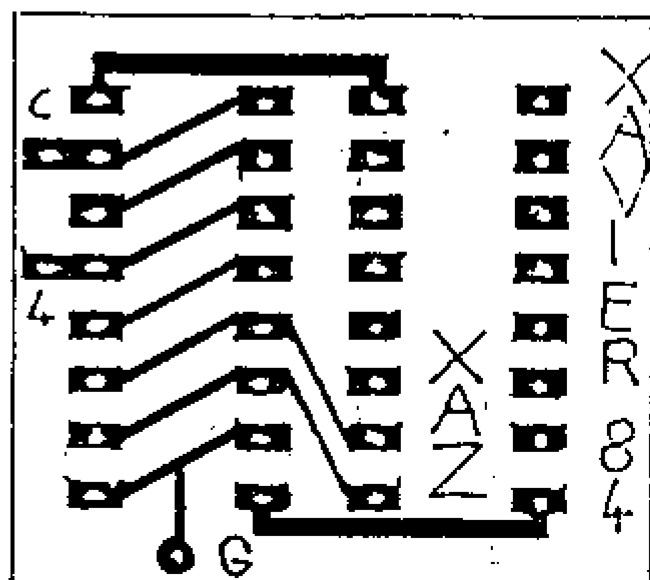
BIJ MIJ FUNCTIONEERT DIT PRINTJE TOT VOLLE TEVREDENHEID MET CHARON OP #400 EN EEN PAAR STATEMENTS OP #C00.

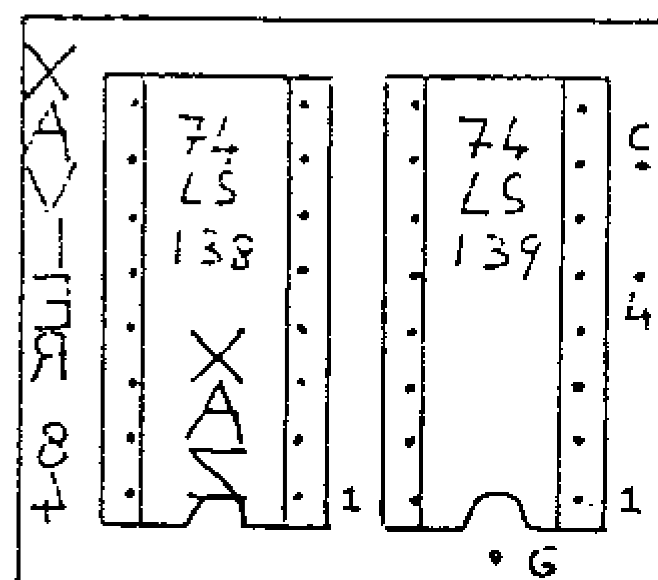
1

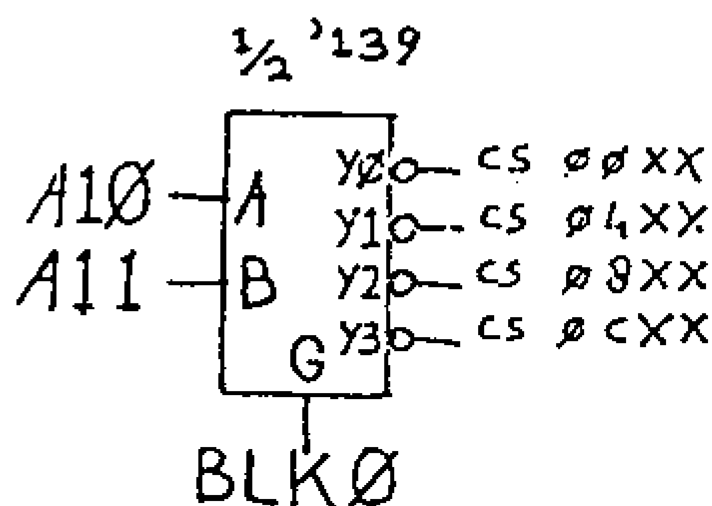

XAVIER 84
XAZ
G

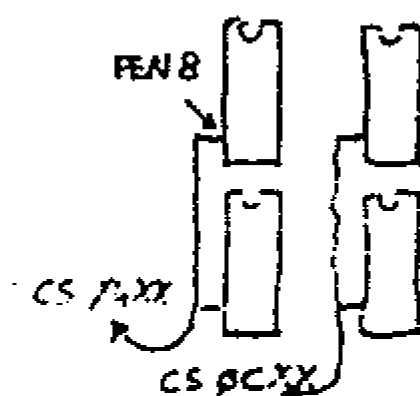

XAVIER 84
LEEG
XAZ
74 LS 139
1
1
G

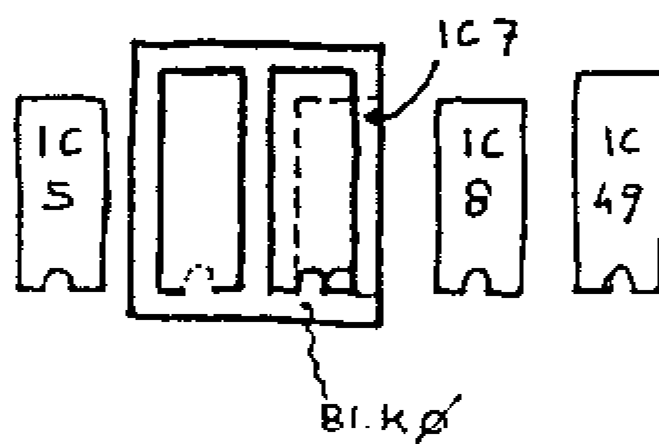
2


C
4
XAVIER 84
XAZ
G


XAVIER 84
74 LS 138
XAZ
74 LS 139
C
4
1
1
G


½ '139
A10
A11
A
B
G
Y0
Y1
Y2
Y3
CS 00XX
CS 04XX
CS 08XX
CS 0CXX
BLK0


PEN 8
CS 04XX
CS 0CXX


IC 7
IC 5
IC 8
IC 49
BLK0

XAZ © 84

CK-SOS

In de ACORN NIEUWS zijn reeds eerder schakelkaart operating systems beschreven. Deze systemen hebben allen hun specifieke voor -en nadelen. Door mij is een SOS ontwikkeld wat ten opzichte van deze systemen een aantal voordelen heeft.

Alle boxen kunnen door elkaar gebruikt worden, omdat steeds bij het wisselen van box de systeemvectoren die naar #AXXX wijzen vervangen worden door een andere vector die bij het aanroepen van deze systeemvector naar de bijbehorende box schakelt.
Zo is het bijvoorbeeld mogelijk het met de P-CHARME in de TXMOD van de JOSBOX te werken, een DISAS of XDUMP uit de josbox m.b.v. het TEXT statement uit de WORDPACK op te slaan, of bij het wegschrijven van file's vanuit de WORDPACK de 1200 BAUD routine uit de josbox te gebruiken.

De vectoren die door het SOS veranderd kunnen worden zijn de BRKVEC, de IRQVEC, de COMVEC, de WRCVEC, de RDCVEC, de LODVEC, de SAVVEC, de BGTVEC en de BPTVEC.

Het SOS heeft wel enige vrije werkruimte nodig i.v.m. het bewaren van de oude systeem vectoren en de bijbehorende chipnr's, hiervoor is het gebied #21C tot #23F gebruikt, dit geeft geen problemen met de P-CHARME, JOSBOX, WORDPACK, CALCROM, TOOLBUG en SOFTTOOL 1+2. Deze ruimte mag absoluut nergens anders voor gebruikt worden, bij problemen met andere boxen kan eventueel naar #9800 worden uitgeweken.

Het is niet nodig boxen in een bepaalde voet van de schakelkaart te plaatsen, omdat het SOS geen eigen commando lijst heeft voor het detecteren van de boxcommando's. Alle boxen worden sequentieel afgezocht totdat het commando is uitgevoerd. Dit werkt uiteraard traag (bij het afzoeken van 5 boxen ongeveer 20 milliseconden). Dit gebeurt echter alleen indien er van box overgeschakeld moet worden.
De filosofie hierachter is n.l. dat de meeste programma's voor een bepaalde box geschreven zijn, is eenmaal een bepaalde box ingeschakeld bijv. P-CHARME door het PROGRAM stat. dan zal deze box ingeschakeld blijven en worden verdere P-CHARME statement's zonder vertraging afgewerkt. Als er nu een onbekend stat. moet worden uigevoerd zullen alle boxen worden afgelopen totdat de P-CHARME weer bereikt wordt.
Het SOS gaat hierna in een eigen interpreter kijken of het stat. herkend wordt.

Het SOS kent de volgende commando's (men kan naar eigen inzicht commando's weglaten of toevoegen) :

LOCK (chipnr.), Zet het SOS vast op een box en reset alle systeem vectoren.

UNLOCK, Maakt een LOCK commando ongedaan.

CHIP (chipnr.), Idem als unlock maar schakelt over naar gespecificeerde box.

PAR, Geeft een lijst van alle basic progs die met REM beginnen. (E. RONDA)

QJ (pag), Vervangt ?18=#(pag);END (A. MARCHAL)

QS (pag), Vervangt ?18=#(pag);END;RUN (A. MARCHAL)

TYPE , Schakelt een nieuwe keyboard en writechar routine in. mogelijkheden zijn autorepeat, fastrepeat, keyboardclicks (piepjes), slowlisting(CTRL-SHIFT), listingstop (SPATIE balk), extra CTRL chars(CTRL-Y graphics printen, CTRL-N weer normaal printen).(gedeeltelijk overgenomen van B. POOT)

Bij het printen van een char op het scherm zijn geen WAIT's meer toegevoegd. Dit geeft een lichte storing op het scherm tijdens het printen, maar werkt wel veel sneller.

CLICK, Schakelt de piepjes van TYPE aan of uit.

TGLBRK, Bij het gebruik van de TYPE read/write routines wordt de BRKVEC en de BASIC ERROR HANDLER niet steeds na het inlezen van een (CR) in de directmode gereset, deze vectoren zijn dus vrij te veranderen. Het is dus b.v. mogelijk na een P-CHARME programma de gebruikte meerdimensionale arrays of functies te printen. Door TGLBRK te gebruiken kan deze functie uit-of weer aan gezet worden.

DIS (start) [,(eind)] MIRAKEL disassembler, verder zelfde syntax als DISAS van de josbox.

MFIND Machinecode find functie.Er zijn drie mogelijkheden:
MFIND XXXX YYYY LDA@#FF (assembler find) zoekt naar de bijbehorende bytes en drukt alle adressen af waar deze voorkomen.
MFIND XXXX YYYY #1234 (byte find) zoekt naar de bytes #34 en #12.
MFIND XXXX YYYY "string" zoekt naar de opgegeven string.

Het systeem is zowel naar #E000 als #1000 (voor de DOS gebruikers) te assembleren en heeft nog voldoende vrije ruimte voor het inbouwen van de BOOTSTRAP routine. Het is nodig de FP ROM te veranderen zodat deze naar #E000 of #1000 springt, zie A.N. 1983 nr.2 blz.74 .

Bij het inschakelen van de ATOM moet het SOS geinitaliseerd worden door op adres #237 (het leesadres voor #BFFF) een nul te schrijven m.b.v. ?#237=0 of door LOCK0;UNLOCK in te typen. Mensen die de BOOTSTRAP hebben ingebouwd kunnen dit natuurlijk makkelijk automatisch doen.
Het hierboven beschreven SOS is zeer gebruikers vriendelijk, het systeem is zondermeer compatibel met de ATOM en werkt zo efficient dat het nauwelijks opvalt, ook vastlopen door verkeerde systeemvectoren indien er van box is overgeschakeld is met dit systeem praktisch niet mogelijk.

M. C. D. R.

-1 BESCHRIJVING MDCR CONTROLLER 2.0

Dit programma is een besturings-programma voor de Mini Digitale Cassette Recorder van PHILIPS, de MDCR moet aangesloten zijn volgends het schema van G.Borgaerts of de beschrijving in het PET bulletin PBE 1980-3;MDCR-PET, hetzij op de eigen (ATOM)VIA of een Extra VIA. Het programma werkt meteen indien men een extra VIA heeft op #BFE0, voor de standaard VIA is slechts een kleine verandering nodig.

De MDCR-controllersource is +/- 16 Kbyte lang en levert een machine code van 3595 bytes. Dit programma kan in een EPROM als uibreiding van het operating-system.

-2 HET PROGRAMMA

Het programma schrijft blokken van 1kbyte op een minicassette, deze blokken hebben elk een naam, de naam van elk blok staat in het eerste blok dat op de band staat. Telkens als er iets op de band geschreven wordt moet dus het eerste blok herschreven worden en wordt er dus teruggespoeld.
Alle blokken die de zelfde naam hebben vormen samen een file die niet aaneengesloten op de band hoeft te staan, deze constructie maakt een optimaal bandgebruik mogelijk, met tevens de mogelijkheid files te herschrijven. Voordat een file weggeschreven wordt, wordt eerst het voorgaande blok geladen en gecontroleerd of deze het juiste bloknr. heeft. Bij het schrijven van een file worden eerst eventuele lege blokken aan het begin van de band geschreven en daarna de blokken aan het einde van de band.
Om te voorkomen dat ook bij het laden van een file steeds vanaf het begin van de band gezocht moet worden, wordt de directory met namen in het RAM-geheugen bewaard, in de standaard versie wordt hier het gestapelde hoge geheugen van #9C00 tot #9FFF gebruikt.

Het wegschrijven van een programma gebeurt met een baudrate van 6000 bits per seconde, er passen dan 50 blokken op een kant van een normale minidatacassette, voor elk blok is in de directory een ruimte van 20 bytes gereserveerd, dit betekent dat de filenaam 19 karakters lang mag zijn.

In de directory staan tevens:

-Op #9FFC het aantal Blokken dat in de directory zit, normaal 50. Indien er bijvoorbeeld maar 49 blokken op de band zouden passen kan dit verholpen worden door in te typen ?#9FFC=49;*MDIR (schrijf de directory opnieuw).

-Op #9FFD de ruimte die per blok in de directory gereserveerd is, normaal 20.

-Op #9FFE een checksum over de directory van #9C00-#9FFD, dit wordt steeds bij de uitvoer van de commando's gecontroleerd en indien er een fout in zit komt de melding DIRECTORY ERROR, dit om

te voorkomen dat er vreemde dingen gebeuren als de directory niet klopt.

-Op #9FFF het bloknr. van het blok waar de band momenteel staat, oftewel waar de controller denkt dat de band staat. Dit adres hoort eigenlijk niet meer bij de directory en wordt ook niet overschreven indien de directory geladen wordt.(de directory is dus eigenlijk 1023 bytes lang.)

-3 FOUTMELDINGEN

Het programma heeft een uitgebreide foutmelding, bij elke fout wordt eerst een bericht afgedrukt en daarna een BRK gegeven, het errornummer is voor alle fouten gelijk en afhankelijk van het adres waarnaar de controller geassembleerd is.
Indien tijdens het laden een van de fouten NO PREAMBLE, NOTHING ON TAPE of INCORRECT LOAD optreedt, spoelt de controller een blok terug en probeert het blok waar het fout ging nogmaals te laden. Als de fout 3 keer achter elkaar optreedt wordt er een foutmelding gegeven.

De fout meldingen zijn:

NO CASSETTE, geen cassette.

END OF TAPE, de cassette is op het einde van de band, terwijl de controller denkt dat hij nog verder kan spoelen, remedie: spoel de band terug.(*REWIND)

NOTHING ON TAPE, er is al meer als 650 msec geen data van de cassette gelezen, deze melding treedt op als de koppen vuil zijn.

NO PREAMBLE, elk block op de band begint met een byte #AA indien dit byte niet als eerste gelezen wordt volgt deze foutmeding.

INCORRECT LOAD, de checksum van het geladen block klopt niet, deze fout komt praktisch nooit voor.

WRITE PROTECT, er zit geen write-enable plugje in de cassette, deze is dus beschermd tegen schrijf-acties.

FILE NOT FOUND, deze fout treedt op bij een poging een niet bestaande file te verwijderen, te herchrijven, of te verlengen.

TAPE FULL, deze foutmelding wordt gegeven indien er niet meer genoeg ruimte op de band meer is om een file weg te schrijven.

DIRECTORY ERROR, voor het uivoeren van de meeste commando,s wordt altijd gecontroleerd of de checksum over de directory nog wel klopt. Indien dit niet zo is dan wordt deze foutmelding gegeven.

SYNTAX ERROR, er zit een fout in het argument van een commando.

EXISTING FILE, treedt op bij een poging een bestaande file te schrijven zonder het *REWRITE commando te gebruiken.

-4 DE COMMANDO INTERPRETER

Het programma heeft een eigen interpreter en commando tabel, welke door het veranderen van de COMVEC beschikbaar wordt. Als het commando niet herkend wordt stuurt het programma door naar #F8EF dit is het normale adres voor de * commando's.

-5 DE COMMANDO.S

Het programma kent de volgende commando's :

*MSAVE "naam" XXXX YYYY ZZZZ
XXXX=STARTADRES FILE
YYYY=EINDADRES FILE
ZZZZ=EXECUTIE ADRES FILE(optional)
Schrijft een file naar de band en spoelt daarna terug om de directory aan het begin van de band aan te passen. Als men alleen *MSAVE "naam" gebruikt dan wordt het basic programma dat door de tekstspace pointer aangegeven word weggeschreven.

*MLOAD "naam" XXXX
XXXX=RELOCATIE ADRES(optional)
Laad een file van de minicassette, indien het een basic file betreft (executie adres C2B2 of 00XX), worden ?18 en TOP automatisch goed gezet.

*DELETE "naam"
Verwijdert een filenaam uit de directory.

*REWIND
Spoel de band terug, de motor van de recorder wordt gestart en het programma komt terug in inputmode, vanaf nu totdat het einde van de band bereikt is wordt om de 65msec met behulp van interrupts die via TIMER 1 (van de #B800 VIA) gegenereerd worden, getest of de band al teruggespoeld is. Deze interrupt constructie wordt ook bij *INIT en *MDIR gebruikt, en werkt alleen indien de commando's vanuit direct-mode ingegeven zijn. Bij gebruik in een BASIC programma wordt gewacht tot de band teruggespoeld is. Tijdens het terugspoelen mag de interruptstatus-bit niet geset worden, en mag de IRQVEC niet veranderd worden.

*INIT
Spoel terug en laad de directory van de band (BLOCK0). Dit commando moet men elke keer bij het inzetten van een andere cassette intypen.

*MCAT
Geeft van elke file alle blokken met start, eind en executie adrssen (drukt alleen de blokken die een naam hebben af).

*FORMAT
Maakt een schone directory en schrijft hem op de band. Dit commando moet gebruikt worden om een lege cassette in gebruik te nemen.

*MRUN "naam" XXXX
Als *MLOAD en LINKT automatisch naar he executie adres. Het is mogelijk basic programma's automatisch te starten indien als executie adres 00XX was opgegeven (een auto run naar een zeropage adres komt nooit voor), dan wordt de textspace-pointer op XX gezet, top goed gezet en een RUN gegeven.

*VERIFY "naam" XXXX
Als *MLOAD maar laadt de file zonder hem in het geheugen te zetten, dit voor controle van een belangerijke file of voor het positioneren van de band.

*DIR
Drukt een lijst van alle filenamen op de band af en geeft het aantal blokken dat deze file in beslag neemt, indien tussen twee files lege blokken staan wordt dit aangegeven met EMPTY.

*REWRITE "naam" XXXX YYYY ZZZZ
Als MSAVE maar voor het herschrijven van een bestaande file.

*APPEND "naam" XXXX YYYY ZZZZ
Als MSAVE maar voor het uitbreiden van een bestaande file, te gebruiken indien een programma uit twee geheugen gedeelten bestaat. Let op bij *MRUN wordt als executie adres dan het executie adres van het laatste blok van de file gebruikt.

*MEM
Drukt het aantal Kbyte af dat nog op de band past.

*MDIR
Bij het sequentieel wegschrijven van file's naar de MDCR is het soms lastig dat er steeds teruggespoeld wordt om de directory aan te passen, dit kan immers ook pas als de laatste file is weggeschreven. indien men tijdens het wegschrijven van een file de CTRL toets ingedrukt houdt, zal de directory nog niet meteen op de band gezet worden, dit kan nu achteraf door *MDIR in te typen.

*NOREW
Na uitvoering van dit commando wordt er na het wegschrijven van een file niet teruggespoeld om de directory te herschrijven, dit heeft de zelfde functie als het ingedrukt houden van de CTRL toets, deze conditie wordt opgeheven door *MDIR te gebruiken (zie ook *MDIR).

*MDCR
Dit commando vervangt de OSSAVE en OSLOAD vectoren door adressen die naar MDCR routines wijzen, hierna kan de MDCR ook in bijvoorbeeld de WORDPACK ROM de CALC ROM of in een data-base programma gebruikt worden. Let op indien nu dus het *SAVE en *LOAD commando gebruikt worden wordt er naar de MDCR toegeschreven en vervalt ook de controle op het reeds bestaan van een file zoals dit door MSAVE, REWRITE en APPEND wel gebeurt, de maximale naamlengte is nu ook maar 13 karakters.

*SYNC
Door dit commando kan men het block nr. waar de band staat synchroniseren met het programma. Dit is te gebruiken indien men een programma van de band wil laden zonder eerst de directory op te halen.

*BLOAD (start block) [,(eind block)]
Met dit commando is het mogelijk per block te laden, dus *BLOAD 1 zal het eerste programmablock van de band laden naar zijn oorspronkelijke adres.

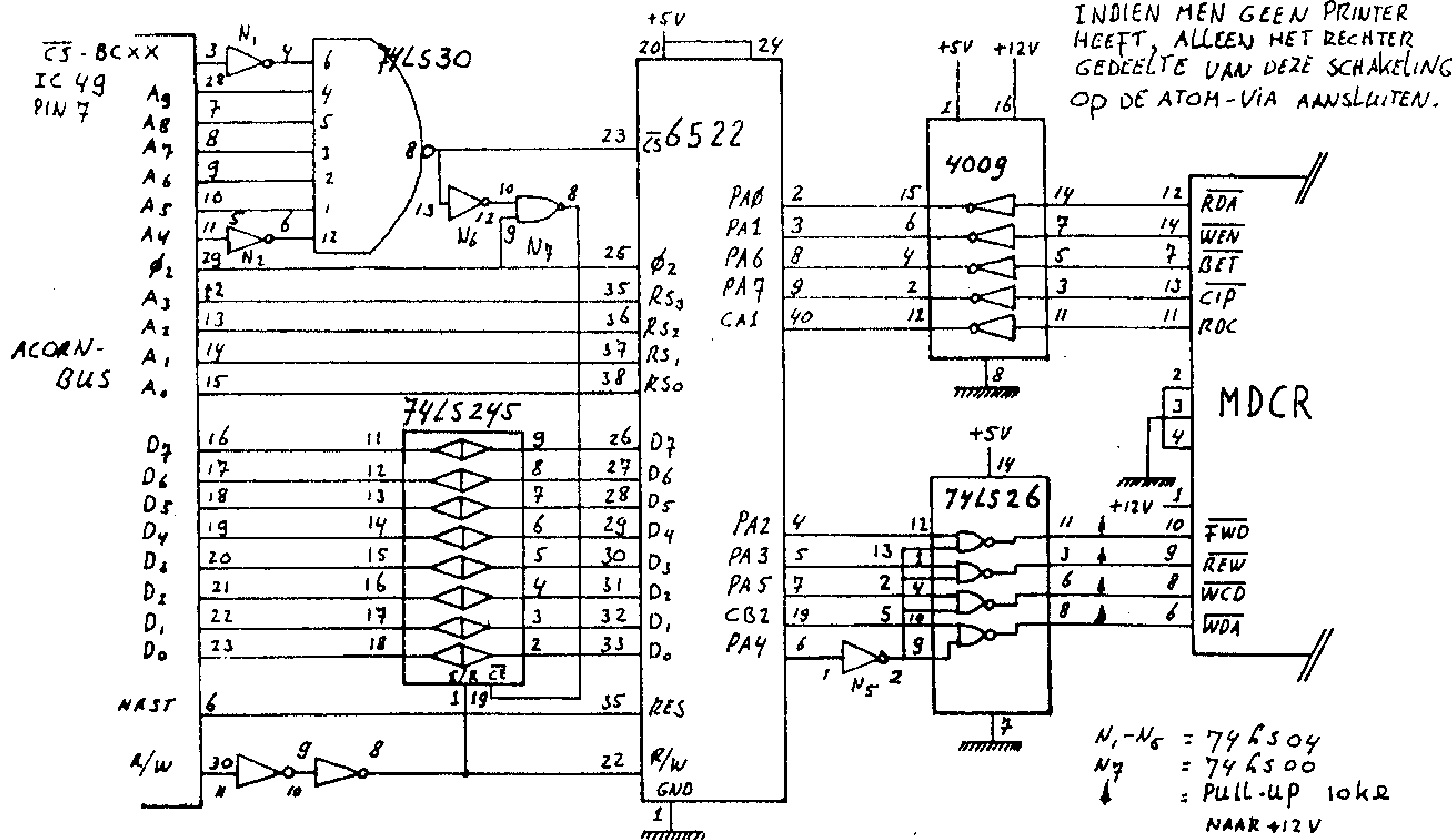
*FILL
Dit commando is handig als om een of andere reden de directory niet van de band geladen kan worden. Het commando schrijft de directory in het geheugen vol met #0D, alle blocks worden vanaf nu in de *MCAT gelist. Op deze manier kan men door naar de start en eindadressen te kijken de verschillende files herkennen en deze met het *BLOAD commando laden.

-6 NOG SNELLER ?

Het is ook mogelijk het programma bij 10000 baud te gebruiken, voor het assembleren van de source dient men deze keuze te maken. De betrouwbaarheid hiervan is nog niet op grote schaal beproefd, in tegenstelling tot het werken met 6000 Baud hetgeen door de fabriek voorgeschreven wordt, en ook al jaren met succes toegepast wordt in andere computers. Dit hangt ook samen met de gebruikte bandkwaliteit. Bij mij werkt het goed, het gebeurt alleen weleens dat een block niet meteen de eerste maal goed gelezen wordt, de tweede maal lukt dit altijd.

Beide systemen zijn wat het laden betreft compatibel, omdat zij dan de gegevens uit de directory gebruiken. Het is dus zonder meer mogelijk 6000 baud en 10000 baud cassettes door elkaar te laden. Het schrijven met 10000 baud werkt alleen goed indien de cassette ook met die snelheid is geformatteerd.

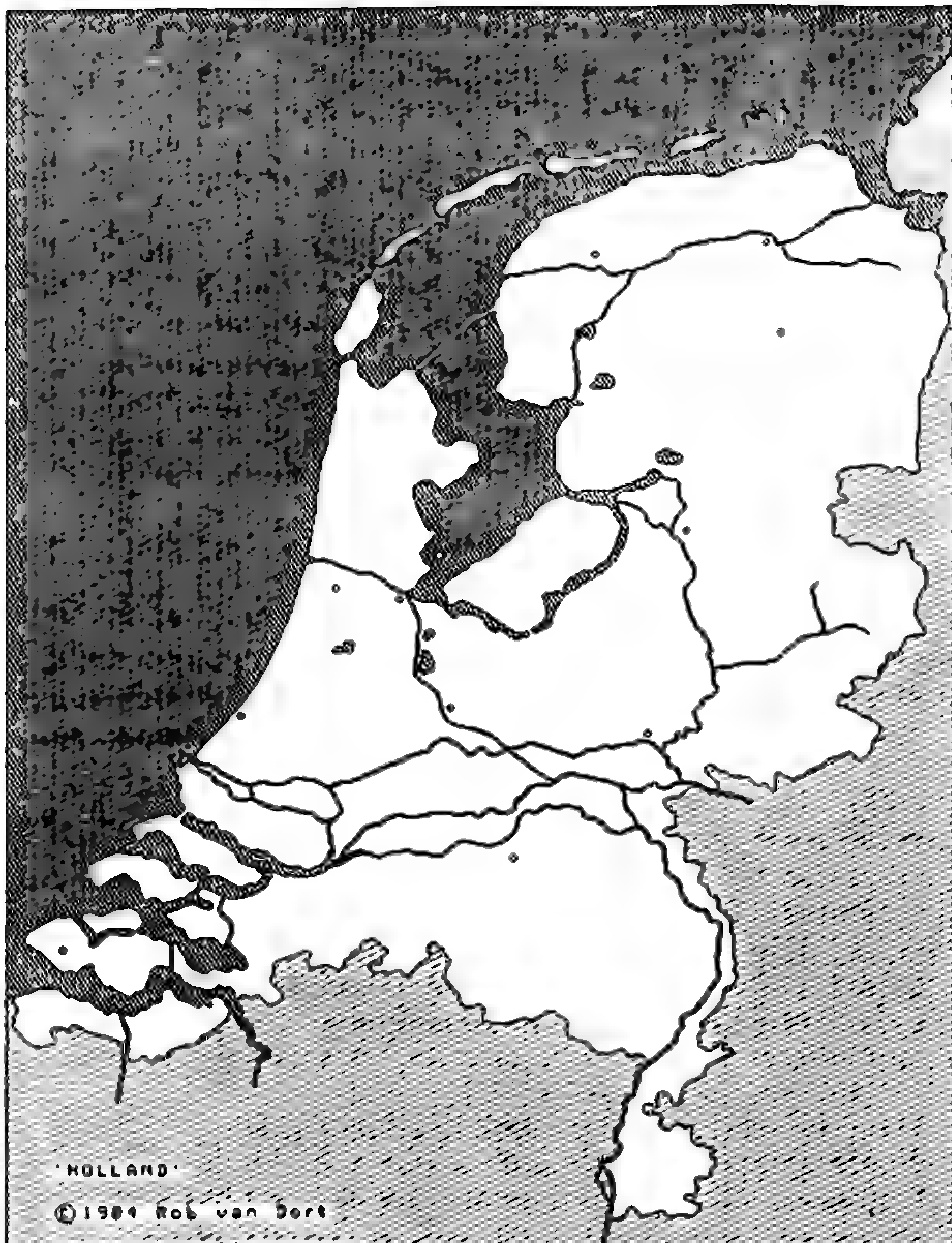
Bij 10kbaud passen er op de cassette 72 blokken, dit houdt in dat de file naam dan ook maar 13 karakters lang mag zijn.



MDCR OP EXTRA VIA - ADRES BFE0 - BFEF

ONTWERP: G. BORGHAERTS / PET-BULLETIN

GET.- C. KWAKERNAAK 7-7-'84


'HOLLAND'
©1984 Rob van Dort
Acorn Atom
micro computer system
actual size 182 x 271 mm
resolution 512 x 768 pixels

Na het lezen van het 1-2-4 MHz artikel uit 1982 werd mijn computer probleemloos tot 2 MHz opgevoerd. Dit blijkt echter bij sommige ATOM's niet mogelijk te zijn. Zo kwam er iemand bij me wiens ATOM op 2 en ook op 1,8 MHz (de halve frequentie van het 3,58 MHz signaal) volkomen stil bleef.
Om uit te vinden waar dat aan lag, werden eerst alle belangrijke en belangrijk lijkende IC's uitgewisseld. Omdat dit geen resultaat had werden alle weerstanden en condensatoren aan een grondige maar tevergeefse controle onderworpen. De conclusie was toen dat de fout in de print moest zitten, en inderdaad waren er enkele minieme verschillen (o.a. spoorbreedtes) waarneembaar.
Na enig experimenteren bleek dat de ATOM in 'standaard' vorm (8+2K) wel op te voeren was. Het monteren van pull-up weerstanden op de 8 datalijnen had het zelfde effect als het weghalen van de (schijnbaar nutteloze) weerstanden R39, R40 en R41, namelijk geen enkele waarneembare activiteit meer bij elk van de voorhanden zijnde kloksignalen tussen 0,5 en 2 MHz.
Uiteindelijk werden er 8 pull-down weerstanden van 4k7 aan de 8 datalijnen gehangen, waarna de ATOM het op alle snelheden (dus ook op 2 MHz) wel goed deed. De aanwezigheid van R39-41 werd hierbij genegeerd zodat op D1-3 twee pull-down weerstanden aanwezig zijn.
De weerstanden werden in dit geval aan de uitgangsbuffer (8304 pennen 1-8) gesoldeerd, maar deze plaats is waarschijnlijk niet kritisch zodat ook andere plaatsen gebruikt kunnen worden.
Een ander punt bij het opvoeren is de processor. Volgens de specificaties gaat de 6502 tot 1 en de 6502A tot 2 MHz. Dit moet echter voor de bezitters van een gewone 6502 geen reden zijn om geen hogere kloksnelheden te proberen, want er zijn wel ATOM's met een gewone processor die ook op 2MHz werken.

Willy Slendebroek

ATOM STRINGFUNCTIES.

Dit programma STRINGFUNCTIONS biedt de gebruiker een 17-tal stringhandling-functies van uiteenlopende aard.
Het programma is geschreven in p-charme formaat en kan binnen een eigen programma worden opgenomen of m.b.v. een ATTACH/EXTERN combinatie elders vanuit het geheugen worden gebruikt (zie A.N. 3/2).
De mogelijkheden van de verschillende stringfuncties zijn zo opgebouwd, dat de functies voor zeer veel toepassingen bruikbaar zijn, bijvoorbeeld een complexe (BASIC) tekstverwerker, of het screenen van user-input (checken op foutieve invoer, analyse van zinsopbouw etc.).

Het programma:

Het programma is in Basic geschreven en heeft daarom als grootste nadeel de lage snelheid. Aan het begin van het programma worden twee variabelen geinitialiseerd, te weten het beginadres van de string-storage-area (B) en de maximale stringlengte (L). Beide variabelen kunt u zelf aanpassen, echter men dient er rekening mee te houden dat de totale string-storage-area 21 maal de max. stringlengte aan bytes in beslag neemt, een goede plaats voor deze opslag is het grafische geheugen. De reden dat elke functie zijn eigen werkgebied krijgt toegewezen ligt hem in het feit dat de functies elkander dan ook kunnen aanroepen. Om geheugen te besparen kan elke functie het zelfde werkgebied toegewezen krijgen, echter men mag binnen een functie dan geen andere functie aanroepen, het gaat dan (bijna) altijd mis, let overigens op dat er reeds enkele functies zijn die al van een andere functie gebruik maken.
De maximale stringlengte is die lengte welke de strings niet zullen overschrijden, alhoewel hiervoor een foutprocedure aanwezig is.

De functies:

LOC(X,Y)   geeft de locatie van substring $Y in string $X, deze functie is bijna identiek aan de functie INSTR (pcharme), echter aangepast aan het in dit programma geldende feit dat niet strings zelf als parameter worden meegegeven, maar de stringadressen, daarnaast geldt dat de eerste locatie in een string altijd positie '0' is.

```
Voorbeeld: $X="STRINGFUNCTIE"
           $Y="FUN"
           LOC(X,Y)=6
```

CONCAT(X,Y)   levert de optelstring van $X en $Y (in die volgorde), de strings worden a.h.w. aan elkander 'geplakt'.

```
Voorbeeld: $X="STRING"
           $Y="FUNCTIE"
           $CONCAT(X,Y)="STRINGFUNCTIE"
```

```
            $Z="nn"
            $REPLAC(X,Y,Z)="STRInnGFUnnCTIES"
```

COMPAR(X,Y) geeft de verschilstring van $X en $Y, beide strings worden vergeleken, en de karakters welke wel in $X maar niet in $Y aanwezig zijn komen achter elkander in de nieuwe string.

```
Voorbeeld:  $X="ATOM STRINGFUNCTIES"
            $Y="AEIOU"
            $COMPAR(X,Y)="TM STRNGFNCTS"
```

RELAT(X,Y) vergelijkt de strings $X en $Y qua lengte en inhoud, indien $X)$Y dan levert de functie '1', is $X=$Y dan '0' en is $X($Y '-1'. Een kortere string wordt als kleiner beschouwd, de vergelijking van de karakters gaat in ascii volgorde.

```
Voorbeeld:  $X="jansen"
            $Y="janssen"
            RELAT(X,Y)=-1
            RELAT((LEFT(X,3),(LEFT(Y,3))=0
```

CONVERT(X,Y,Z) converteert alle de karakters van substring $Y in string $X m.b.v. Z (om precies te zijn een EOR met Z).

```
Voorbeeld:  $X="STRINGFUNCTIES"
            $Y="STRING"
            $CONVERT(X,Y,#20)="ATOM stringFUNCTIES"
```

RELOC(X,Y,Z) reloceert de posities van substrings binnen $X a.d.h.v. vergelijkings-string $Y en delimiter-karakter Z.

```
Voorbeeld:  $X="*aa*bb*ccc*dd*e"
            $Y="    *    *    *    *    *    "
            $RELOC(X,Y,CH"*")="    aa   bb   ccc dd   e   "
```

Let er overigens wel op dat aan het eind van de vergelijks-string voldoende ruimte is gecreeerd om ook de laatste substring goed te kunnen reloceren, de relocatie stopt namelijk zodra een van de string $X of $Y is doorlopen.

TRANSLAT(X,Y,Z) vertaal de string $X met behulp van taalstring $Y naar vertaalstring $Z. Deze functie bepaalt voor elk karakter in de string $X de positie in $Y, en vormt een vertaalde string door het met deze positie overeenkomende karakter uit de vertaalstring $Z te vervangen, indien het karakter niet in $Y aanwezig is, komt een spatie in de vertaalde string.

```
Voorbeeld:  $X="ATOM STRINGFUNCTIES"
            $Y="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
            $Z="ZYXWVUTSRQPOMNLKJIHGFEDCBA"
            $TRANSLAT(X,Y,Z)="ZGLN HGIRMTUFMXGRVH"
            $TRANSLAT(TRANSLAT(X,Y,Z),Z,Y)="ATOM
STRINGFUNCTIONS"
```

Wellicht ten overvloede wijs ik er op dat men in de functie geen STRING als paramter kan meegeven, alleen een stringadres, een functie:

```
PRINT $COMPAR(X,"AEIOU")
```

is dus niet toegestaan.

```
100 PROGRAM STRINGFUNCTIONS
110
120 B=#9000;REM string storage area
130 L=64    ;REM max stringlength
140          REM size of string storage area = B+21*L
150 FUNCTION LOC(X,Y)
160  IF LEN(Y))LEN(X) ERROR(1)
170  LOC=INSTR($X,$Y)-1
180 FEND
190
200 FUNCTION CONCAT(X,Y)
210  IF LEN(X)+LEN(Y))L ERROR(2)
220  $B=$X
230  $B+LEN(X)=$Y
240  CONCAT=B
250 FEND
260
270 FUNCTION LEFT(X,Y),A
280  IF Y(1 ERROR(3)
290  IF Y)LEN(X) ERROR(4)
300  A=B+L
310  $A=$X
320  Y?A=#D
330  LEFT=A
340 FEND
350
360 FUNCTION RIGHT(X,Y),A
370  IF Y(1 ERROR(3)
380  IF Y)LEN(X) ERROR(4)
390  A=B+2*L
400  $A=$X
410  IF Y(LEN(X) $A=$X+LEN(X)-Y
420  RIGHT=A
430 FEND
440
450 FUNCTION MID(X,Y,Z),A
460  IF Z(1 ERROR(3)
470  IF Y)LEN(X) OR Y(0 ERROR(5)
480  A=B+3*L
490  ?A=#D
500  IF Y(LEN(X) $A=$X+Y
510  Z?A=#D
520  MID=A
530 FEND
540
```

```
 550 FUNCTION PDELET(X,Y,Z),A
 560  IF Z<1 ERROR(3)
 570  IF Y>LEN(X) OR Y<0 ERROR(5)
 580  A=B+4*L
 590  $A=$X
 600  ?(A+Y)=#D
 610  $A=$CONCAT(A,(X+Y+Z))
 620  PDELET=A
 630 FEND
 640
 650 FUNCTION SDELET(X,Y),A
 660  A=B+5*L
 670  $A=$X
 680  DO
 690   $A=$PDELET(A,(LOC(A,Y)),LEN(Y))
 700  UNTIL LOC(A,Y)=-1
 710  SDELET=A
 720 FEND
 730
 740 FUNCTION INSERT(X,Y,Z),A
 750  IF LEN(X)+LEN(Y)>L ERROR(2)
 760  IF Z>LEN(X) OR Z<0 ERROR(5)
 770  A=B+6*L
 780  $A=$X
 790  $(A+L)=$(A+Z)
 800  $(A+Z)=$Y
 810  $A=$CONCAT(A,(A+L))
 820  INSERT=A
 830 FEND
 840
 850 FUNCTION ROTAT(X,Y,Z),I,J,A
 860  IF Y<0 OR Y>LEN(X) ERROR(5)
 870  IF Y+Z>LEN(X) ERROR(4)
 880  IF Z<1 ERROR(3)
 890  A=B+8*L
 900  $A=$X
 910  I=0
 920  DO
 930   J=?(A+I+Y)
 940   ?(A+I+Y)=?(A+Y+Z-I-1)
 950   ?(A+Y+Z-I-1)=J
 960   I=I+1
 970  UNTIL I=Z/2
 980  ROTAT =A
 990 FEND
1000
1010 FUNCTION PSPLIT(X,Y,Z),A
1020  A=B+9*L
1030  $A=$LEFT(X,Y)
1040  PSPLIT=A
1050  $Z=$(X+Y)
1060 FEND
```

```
1070
1080 FUNCTION SPLIT(X,Y,Z),I,A
1090  A=B+10*L
1100  I=LOC(X,Y)
1110  $A=$PSPLIT(X,I,Z)
1120  SPLIT=A
1130 FEND
1140
1150 FUNCTION REPLAC(X,Y,Z),I,J,K,A
1160  IF LEN(X)-LEN(Y)+LEN(Z))L ERROR(2)
1170  IF LOC(Z,Y)()-1 THEN ERROR(6)
1180  A=B+11*L;I=A+L;J=I+L;K=J+L
1190  $J=$X
1200  DO
1210   $I=$SPLIT(J,Y,K)
1220   $K=$RIGHT(K,(LEN(K)-LEN(Y)))
1230   $K=$CONCAT(Z,K)
1240   $J=$CONCAT(I,K)
1250  UNTIL LOC(J,Y)=-1
1260  $A=$J
1270  REPLAC=A
1280 FEND
1290
1300 FUNCTION COMPAR(X,Y),I,J,A
1310  A=B+15*L;J=A+L
1320  $J=""
1330  FOR I=0 TO LEN(X)
1340   $A=$MID(X,I,1)
1350   IF LOC(Y,A)=-1 $J=$CONCAT(J,A)
1360  NEXT I
1370  COMPAR=J
1380 FEND
1390
1400 FUNCTION RELAT(X,Y),I,J,A
1410  I=-1;J=9
1420  DO
1430   I=I+1
1440   XIF X?I=#D AND Y?I=#D;J=0
1450    ELSE XIF X?I)Y?I OR Y?I=#D J=1
1460     ELSE IF X?I(Y?I OR X?I=#D J=-1
1470  UNTIL J()9
1480  RELAT=J
1490 FEND
1500
1510 FUNCTION CONVERT(X,Y,Z),I,J,A
1520  A=B+17*L
1530  J=LOC(X,Y)
1540  IF J=-1 THEN ERROR(6)
1550  $A=$X
1560  FOR I=J TO J+LEN(Y)-1
1570   A?I=(A?I):Z
1580  NEXT I
1590  CONVERT = A
1600 FEND
```

```
1610
1620 FUNCTION RELOC(X,Y,Z),I,J,A
1630  A=B+18*L
1640  I=0;J=0
1650  DO
1660   XIF X?I=Z
1670    I=I+1
1680    WHILE Y?J()Z
1690     A?J=32
1700     J=J+1
1710    WEND
1720   ELSE A?J=X?I;J=J+1;I=I+1
1730  UNTIL X?I=#D OR Y?J=#D
1740  A?J=#D
1750  RELOC=A
1760 FEND
1770
1780 FUNCTION TRANSLAT(X,Y,Z),I,J,K,A
1790  I=0;A=B+19*L;K=A+L
1800  DO
1810   J=LOC(Y,MID(X,I,1))
1820   XIF J=-1 OR J)LEN(Z)
1830    K?I=32
1840   ELSE K?I=Z?J
1850   I=I+1
1860  UNTIL X?I=#D
1870  K?I=#D
1880  TRANSLAT=K
1890 FEND
1900
1910 PROC ERROR(X)
1920  CASE X OF
1930(1)  PRINT"SUBSTRING TOO LARGE"'
1940(2)  PRINT"STRING TOO LARGE"'
1950(3)  PRINT"NO OF CHARS TOO SMALL"'
1960(4)  PRINT"NO OF CHARS TOO LARGE"'
1970(5)  PRINT"WRONG CHARACTERPOSITION"'
1980(6)  PRINT"SUBSTRING NOT FOUND"'
1990  CEND
2000  BEEP50,25
2010  END
2020 PEND
```

S P E L V O O R A C O R N - A T O M + A X R - 1

DOBBELLATTEN

Een spel waar van de spelers zowel reactievermogen als besluitvorming vereist wordt.
Maximaal zes spelers proberen om de minste totaalpunten bij elkaar te dobbelen en te combineren.
Het spel kan gespeeld worden op de Acorn-Atom voorzien van een AXR-1 toolkit en is zeer geschikt als gezelschapsspel.

Het spel

Bij aanvang van het spel hebben we negen genummerde vakjes boven in het scherm staan.
De eerste speler stopt nu met een druk op de spatiebalk de eerste dobbel-lat (een platte dobbelsteen)en het gedobbelde cijfer blijft zichtbaar.
Daarna wordt automatisch de tweede dobbellat gestart, welke door de speler weer gestopt kan worden.
Met de uitgekomen cijfers kan de speler nu vakjes gaan sluiten, en dit herhaalt zich totdat de speler geen enkel vakje meer kan sluiten. Het totaal van de opengebleven vakjes wordt nu bij elkaar opgeteld en bij het totaal van de speler bijgeteld. Een uitkomst van Ø is dus het mooiste, want dat betekent dat alle vakjes gesloten werden door de speler.
De gedobbelde cijfers kunnen, indien mogelijk ook gesplitst worden, maar dit wordt nog uitgelegd.
Als er geen keuze bestaat voert de computer automatisch de énige keuzemogelijkheid uit.
Heeft een speler in een ronde het maximum overschreden, dan komt er achter zijn score een # teken te staan en is hij voor de rest van de ronde uitgeschakeld.
Bij zijn uiteindelijke totaalscore worden nu echter elke keer dat andere spelers nog niet aan het maximum zitten 50 extra strafpunten toegevoegd.
Dit zal straks met een voorbeeld verduidelijkt worden in een spelverloop-notatie.
De winnaar is diégene die na alle gespeelde rondes de minste punten heeft.

Aanvang van het spel

De computer heeft een aantal gegevens van ons nodig voordat het spel kan beginnen.
Wij dienen dan ook de navolgende zaken via het toetsenbord in te voeren.

- Aantal spelers — Kan lopen van 1 t/m 6
- Moeilijkheidsgraad — Keuze van 0 t/m 4
- Aantal rondes dat men wil spelen.

Moeilijkheidsgraad

De cijfers 1 t/m 9 samen opgeteld geeft 45. Als je nu een maximum van 50 invoert, kan je in de eerste ronde bij je eerste beurt weinig gebeuren, want je kunt jezelf dan niet kapotspelen.
Moeilijker wordt het daarom als het maximum slechts 10 bedraagt.
Je mag dan slechts 9 punten over hebben om jezelf niet meteen kapot te spelen.
Hoe hoger het maximum is ingesteld des te langer kan 1 speelronde dan duren.
Spannender wordt het als je zoveel mogelijk weg moet dobbelen per serie.

Om nu zelf dat maximum te bepalen kun je dit van tevoren volgens bijgaand lijstje invoeren.

Keuze	Maximum
0	10
1	20
2	30
3	40
4	50

Het splitsen

Splitsen betekent dat je met bijvoorbeeld de gedobbelde getallen 3 en 6 zelf kunt bepalen of je deze bij elkaar optelt of apart gebruikt om dan twee hokjes te sluiten. Dit wordt je alleen gevraagd door de computer als de mogelijkheid echt aanwezig is.
Aan de hand van een paar voorbeeldjes zullen we eens kijken hoe dat werkt.

Steeds worden de overgebleven vakjes getoond en de vakjes die dicht zijn worden gemerkt met een $.
De getallen in de vierkantjes geven de gedobbelde getallen weer.
Daaronder vinden we de twee mogelijkheden die we hebben en het gevolg van splitsen of niet splitsen bij die situatie.

```
A.    1 2 3 4 5 6 7 8 9        [6]  [3]

      1 2 $ 4 5 $ 7 8 9        hier is dus gesplitst
      1 2 3 4 5 6 7 8 $        en hier dus niet

B.    1 2 $ 4 5 $ 7 8 9        [1]  [1]

      $ 2 $ 4 5 $ 7 8 9        hier is dus gesplitst
      1 $ $ 4 5 $ 7 8 9        en hier dus niet

C.    $ 2 $ 4 5 $ 7 8 9        [3]  [5]

      $ 2 $ 4 $ $ 7 8 9        hier is dus gesplitst
      $ 2 $ 4 5 $ 7 $ 9        en hier dus niet
```

In voorbeeld A. konden we kiezen tussen 1. 6 en 3
2. alleen de 9 = (6 + 3)

In voorbeeld B. konden we kiezen tussen 1. één 1
2. of 1 + 1 dus de 2

In voorbeeld C. konden we kiezen tussen 1. alleen de 5
2. of 3 + 5 dus de 8.

Spelverloop

Enkele regels welke tijdens het spel toegepast worden door het programma zullen we nog even kort toelichten.

- als er slechts 1 speler binnen 1 ronde nog beneden het maximum zit als alle spelers gespeeld hebben, is de ronde afgelopen en krijgen alle spelers die kapot zijn 50 extra strafpunten.

- als alle spelers zich in de ronde kapot gespeeld hebben en alle spelers gespeeld hebben is er geen rondewinnaar en krijgen alle spelers 50 extra strafpunten.

- als een speler niet meer verder kan spelen worden de totalen van alle spelers zichtbaar op het scherm met de ronde-uitslag.

- na iedere ronde krijgen we een totaaloverzicht van alle spelers en hoe zij er met hun totaalscore voor staan.

- tijdens het spel staat het nummer van de speler die aan de beurt is, invers middenboven de twee dobbellatten.

- de rondeteller geeft aan hoeveel rondes er totaal gespeeld worden en met welke ronde men momenteel bezig is.
  B.V.: stel er staat :   Ronde : 32   dan betekent dit dat ronde 2 aan de gang is en dat er totaal 3 rondes gespeeld worden.

Hieronder volgt een overzicht van een spel dat gespeeld werd met 6 spelers, maximum was ingesteld op 20 en er werden 4 rondes gespeeld.

J.A.F.M. van Eldik.

NUMMERS VAN DE SPELERS	RONDE 1			TOTALEN NA 1e RONDE	RONDE 2			TOTALEN NA 2e RONDE	RONDE 3			TOTALEN NA 3e RONDE	RONDE 4			EINDSCORE NA 4 RONDES
1	00	00	00	00	27	#	#	127	09	30	#	207	17	25	#	282
2	21	#	#	121	09	26	#	197	04	24	#	271	07	33	#	354
3	21	#	#	121	21	#	#	242	21	#	#	363	20	#	#	483
4	15	49	#	99	05	19	19	118	20	#	#	238	11	47	#	335
5	00	34	#	84	22	#	#	206	31	#	#	337	04	04	04	341
6	12	32	#	82	16	42	#	174	19	51	#	275	33	#	#	408

```
 10 P.$12;?#E1=0;@=0;G=1;C=#FF;T=0
 20 PRINT"        **** dobbelspel ****"''
 30 PRINT"MAXIMAAL ZES SPELERS PROBEREN OM"
 40 PRINT"DE  MINSTE  PUNTEN BIJ ELKAAR TEDOBBELEN."'
 50 PRINT"DIT LUKT ALLEEN DOOR OP DE GOEDEMOMENTEN  DE JUISTE"
 60 PRINT" BESLISSINGENTE NEMEN."'
 70 PRINT"DE REACTIESNELHEID VAN DE SPELERHELPT HEM OM OP"
 80 PRINT" HET GOEDE MOMENT"
 90 PRINT"DE DOBBELBALKEN TE STOPPEN EN ZOHET OVERGEBLEVEN"
100 PRINT" TOTAAL  VAN  DECIJFERS TE VERKLEINEN."''
110 PRINT"                       drukopeentoets"
120 LINK #FFE3;PRINT$12;?#E1=0
130 PRINT"AANTAL SPELERS (MAXIMAAL 6) ?"''
140 LINK #FFE3;KEY M;M=M-#30;IF M>6 OR M<1 THEN RUN
150 PRINT"MOEILIJKHEIDSGRAAD  (0-4) ?"''
160 LINK #FFE3;KEY Z;Z=(Z-#30)*10+10
170 IF Z<10 THEN GOTO 150
180 PRINT"HOEVEEL RONDES TE SPELEN  ?"
190 LINK #FFE3;KEY I;I=I-#30;PRINT$12;?#E1=0
200 DIM SS(M),BB10,QQ(M)
210 FOR X=1 TO M;QQX=0;SSX=0;NEXT
220 BB1=#8023;BB2=#8026;BB3=#8029;BB4=#802C;BB5=#802F
230 BB6=#8032;BB7=#8035;BB8=#8038;BB9=#803B;BB0=#8000;?BB0=#20
240 FOR P=1 TO I
250 PRINT$12;?#E1=0
260 PRINT $30''''''''''''';?BB0=#20
270 PRINT" MAXIMUM:"Z"          RONDE:"I,P;?BB0=#20
280 GOSUB j;GOSUB d;GOSUB f
290p N=1
300q IF SSN<Z;DO;;GOSUB u;UNTIL G=0;GOSUB y
310 IF SSN>=Z THEN QQN=QQN+50
320 N=N+1;G=1
330 IF N<=M THEN GOTO q
340 Q=0
350 FOR U=1 TO M
360 IF SSU>=Z THEN Q=Q+1
370 NEXT U
380 IF Q=M OR Q=M-1 THEN GOSUB r;GOTO 400
390 GOTO p
400 NEXT P
410 PRINT"     drukopeentoets";LINK #FFE3;RUN
420a ?#818F=N+#B0;?#816F=#43
430b B=0;FOR X=1 TO 14;A?X=C;NEXT X;FOR Y=1 TO 12 STEP 2
440 B=B+1;WAIT;A?(Y+1)=B+176;A?Y=C;A?(Y-1)=C;IF B=6 THEN Y=12
450 KEY D;IF D=#20 THEN Y=12;IF H=1 THEN E=B;NEXT;RETURN
460 IF D=#20 THEN Y=12;IF H=2 THEN F=B;NEXT;RETURN
470 NEXT Y;GOTO b
480c IF H=1 THEN A=#81C0
490  IF H=2 THEN A=#81D0
500  FOR X=1 TO 14;A?X=C;NEXT;RETURN
510d FORS=#8001 TO #801D;?S=#23;NEXT
520  FORS=#8021 TO #803D;?S=#23;NEXT;RETURN
530f FORS=#8041 TO #805D;?S=#23;NEXT
540  V=#B1;FOR S=#8023 TO #803C STEP 3;?S=V;V=V+1;NEXT;RETURN
550e K=0;L=0
560 IF ?BB(E+F)<>#23 AND E+F<10 THEN K=1
```

```
570 IF ?BBE()#23 OR ?BBF()#23 THEN L=1
580 IF K+L=0 THEN G=0:RETURN
590 IF K+L=2 THEN GOTO g
600 IF K=1   THEN GOTO h
610iGOSUB m:?BBE=#23:?BBF=#23:RETURN
620hGOSUB n:?BB(E+F)=#23:RETURN
630gPRINT "    WILT U SPLITSEN  ?"'
640 LINK #FFE3:KEY D
650 IF D=69  OR D=74  THEN GOTO i
660 GOTO h
670j A=#81C0:GOSUB k:A=#81D0:GOSUB k:RETURN
680k W=A-#20:S=A+#20:FOR X=0 TO 14:W?X=#43:S?X=#70:A?X=C:NEXT
690 ?#81AF=#7F:?#81EF=#70:?#81CF=#7F:?#818E=#5D:?#8190=#6E
700 ?#816F=#43:RETURN
710l T=0:FOR X=1 TO 10
720  P."    uw beurt is voorbij"$13
730  P."    UW BEURT IS VOORBIJ"$13
740  NEXT X:PRINT ''
750  P. "    DRUK NU OP EEN TOETS"
760  LINK #FFE3:PRINT$13$11$11$11$11
770 FOR S=#8060 TO #813F:?S=#20:NEXT
780  FOR X=1 TO 9
790  IF ?BBX()#23 THEN T=T+X
800  NEXT:RETURN
810m FOR X=1 TO 12
820 IF ?BBF()#23 THEN ?(BBF+#20)=#9E:WAIT:WAIT
830 IF ?BBE()#23 THEN ?(BBE+#20)=#9E:WAIT:WAIT
840 IF ?BBF()#23 THEN ?(BBF+#20)=#1E:WAIT
850 IF ?BBE()#23 THEN ?(BBE+#20)=#1E:WAIT
860 NEXT:RETURN
870n FOR X=1 TO 10:?(BB(E+F)+#20)=#9E:WAIT:WAIT
880 ?(BB(E+F)+#20)=#1E:WAIT:WAIT:NEXT:RETURN
890uH=1:GOSUB c:GOSUB a:PRINT $30''''
900 PRINT"    UW GETAL(LEN):  "E" "
910 H=2:GOSUB c:GOSUB a:PRINT" EN "F'':GOSUB e
920 IFG()0 THEN GOSUB y:RETURN
930 IFG=0 THEN GOSUB l
940 SSN=SSN+T:GOSUB j:GOSUB f:?#818F=#7F
950 FORX=1 TO M
960 PRINT"   SPELER "X":   "
970 IFSSX(10 THEN PRINT"0"
980 PRINTSSX"  PUNTEN":IF SSX)=Z  THEN PRINT "  #"
990 PRINT':NEXT
1000 PRINT"      druknuopeentoets":LINK #FFE3:RETURN
1010yFOR X=#8060 TO #815C:?X=#20:NEXT
1020 FOR H=1 TO 2:GOS.c:NEXT:RETURN
1030r FOR X=1 TO M
1040  QQX=QQX+SSX
1050  SSX=0
1060  NEXT X
1070  P.$12:?#E1=0
1080 PRINT"UITSLAGEN NA RONDE  "P''
1090  FOR X=1 TO M
1100 PRINT"SPELER "X"  TOTAAL: "QQX"  PNT."''
1110 NEXT
1120 LINK #FFE3:RETURN
```

Met de komst van het programma 'sourcemaker' is het leven er 'n stuk gemakkelijker op geworden. Een komplimentje voor de makers van dit programma, t.w. Arie Marchal en Ed Ronda, is hier dan ook zeker op z'n plaats. Bij deze dus.
Er is echter een foutje in het programma geslopen. Draai het programma maar eens voor het assemblergedeelte #F500-#F540. In dit gebied zit de string "OUT OF RANGE:<LF> <CR>". Als sourcemaker klaar is en we bekijken onze zojuist gemaakte source van dit gebied dan zien we het probleem. Sourcemaker zet de HELE string over naar het programma met als gevolg dat de <CR> als 'gewone' <CR> gezien wordt wanneer het programma klaar is. Het resultaat zien we in listing 1. Het probleem is dus dat de controle-karakters ook in stringvorm worden overgezet. Liever zouden we dat hebben in een vorm van !#..=#....; In sourcemaker is al een routine aanwezig die dergelijke regels aflevert(voor het maken van de tabel). Aan de hand van deze routine heb ik de 'string'-routine veranderd. Deze zorgt er nu voor dat alle controle-karakters in bovenstaande vorm worden afgeleverd.Zie voor het resultaat listing 2.Als er controle-karakters tussen 'gewone' karakters instaan wordt de string in gedeeltes gebroken.Een voorbeeld hiervan zien we in listing 3.In regel 120 zien we dat er een byte extra wordt meegenomen als er een oneven aantal controle-karakters in een string staat, dit wordt echter weer opgeheven door P wel met het goede aantal op te hogen (in ons geval met 1).
Verder heb ik in het programma nog wat kleine dingetjes veranderd zodat de source die aangemaakt wordt iets leesbaarder is.
De nieuwe sourcemaker zit in het programma-archief.

LISTING 1:oude sourcemaker

```
120SBC@#00;BEQLL6;CMP@#FF;BEQLL5
130;:LL4;;JSRLL7
140];$P="OUT OF RANGE:

8763P=P+L.P;[
150STY#67;BMILL3
```

LISTING 2:nieuwe sourcemaker

```
120 SBC@#00;BEQLL6;CMP@#FF;BEQLL5
130:LL4 JSRLL7
140];$P="OUT OF RANGE:";P=P+L.P
150 P!0=#D0A;P=P+2
160[
170 STY#67;BMILL3
```

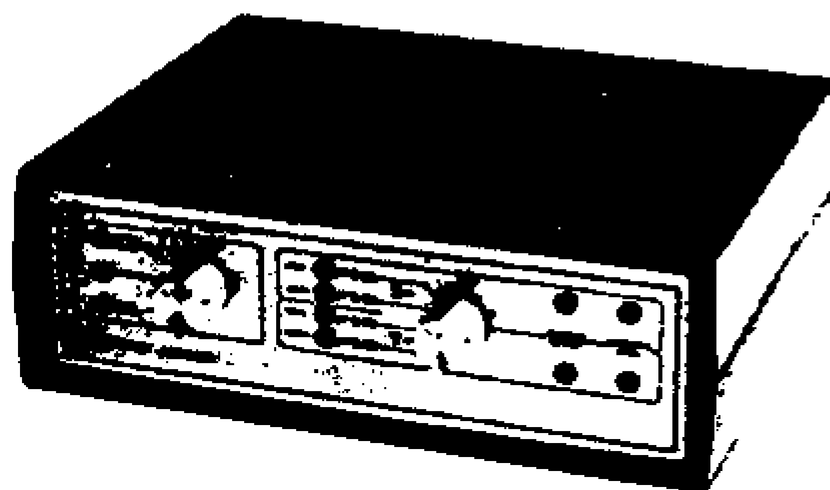
LISTING 3:demo CTRL-karakters

```
110:LL0 NOP;JSRLL1
120[;P!0=#440C;P=P+1
130 $P="DIT IS ";P=P+L.P
140 P!0=#B0B;P!2=#B0B;P=P+4
150 $P="EEN DEMONSTRATIE";P=P+L.P
160 P!0=#A0A;P!2=#A0A;P!4=#D0A;P=P+6
170[
180 NOP;NOP;NOP;NOP
```

De string was:
"<FF>DIT IS <VT,VT,VT,VT>EEN DEMONSTRATIE<LF,LF,LF,LF,LF,CR>"

Uitgeversmij Elektuur b.v.
postbus 75
6190 AB Beek (L)
Bourgognestraat 13a
6191 HX Beek (L)
Nederland
tel: 04402-74200
telex: 56617

elektuur



Unieke mogelijkheid voor computer-hobbyisten!

DIREKT-GEKOPPELDE MULTI-STANDAARD-MODEM VOOR ZELFBOUW MET OFFICIELE PTT-GOEDKEURING

Elektuur, het internationale maandblad voor elektronica, komt in haar septembernummer met een Europese primeur! In deze uitgave wordt namelijk een bouwbeschrijving gepubliceerd van de "Telektor", een direkt-gekoppelde modem waarmee computers via de telefoonlijn met elkaar kunnen kommuniceren. Het unieke van dit zelfbouwprojekt zit 'm in de mogelijkheid om elke opgebouwde "Telektor" door Elektuur te laten keuren, waarna het apparaat wordt voorzien van een officiële PTT-goedkeuringssticker. Een modem die direkt aan het telefoonnet is gekoppeld móét namelijk aan de door de PTT gestelde eisen voldoen èn een PTT-goedkeuring hebben, anders is zo'n direkte koppeling niet toegestaan!

Om de nabouwzekerheid en de specifikaties te kunnen garanderen heeft Elektuur voor deze gelegenheid een bouwpakket samengesteld, waarin alle voorgeschreven onderdelen verenigd zijn: een speciaal IC, lijntrafo, dubbelzijdige print, kastje, frontplaat, speciale telefoonsteker en alle andere "gewone" onderdelen. De prijs van dit bouwpakket bedraagt f 749,—, inklusief keuringskosten. Nadere gegevens hierover vindt u in het septembernummer van het maandblad Elektuur.

Voor de technisch geïnteresseerden nog enkele gegevens van de "Telektor":

- Keuze uit twee standaards:
  V.21 – 300 baud full-duplex
  V.23 – 1200/75 baud split speed full-duplex
- Auto-answer-faciliteit. De modem kan data-boodschappen aannemen als niemand thuis is.
- Twee konnektoren aanwezig voor de koppeling met de computer:
  RS 232-konnektor met V.24-signalen
  Aangepaste konnektor voor TTL-nivo's
- Re-speeder-schakeling voor snelheidsaanpassing van het 75 baud-kanaal bij V.23-mode.
- Zend- en ontvanggedeelte plus de filters zijn digitaal opgebouwd (alles zit in één IC), zodat er geen afregelpunten zijn.

Uitgeversmij. Elektuur B.V., Postbus 75, 6190 AB Beek (L), tel.: 04402-74200.

```
 10 REM  DIAGNONAAL, J. SIEGERS
 20 REM TEKENT VEELHOEK MET DIAGONALEN
 30 IN."AANTAL ZIJDEN "A
 40 CLEAR4
 50 DIM XX(A),YY(A)
 60 I=0
 70 FOR N=0 TO 360 STEP (360/A)
 80    XX(I)=128+%(80*SINRAD N)
 90    YY(I)=100+%(80*COSRAD N)
100   I=I+1
110 NEXT N
120 FOR N=0 TO (I-2)
130    FOR M=N TO (I-2)
140       MOVE XX(N),YY(N)
150       DRAW XX(M),YY(M)
160    NEXT M
170 NEXT N
180 END
```

```
 10 REM flat
 20 CLEAR4
 30 C=190; Y=256; Z=100
 40 FOR B=60 TO 66
 50    FOR D=0 TO 80 STEP 10
 60       MOVE B,(D+Z)
 70       E=40*D/B
 80       G=66*D/(Y-B)
 90       DRAW A,(E+Z); MOVE B,(D+Z): DRAW C,(G+Z)
100       MOVE B,(D+Z); DRAW B,(Z-D)
110       DRAW A,(Z-E); DRAW A,(Z+E)
120       MOVE B,(Z-D); DRAW C,(Z-G); DRAW C,(Z+G)
130    NEXT D
140    D=80
150    FOR N=A TO B STEP 10
160       U=N*D/B
170       MOVE N,(Z+U); DRAW N,(Z-U)
180    NEXT N
190    FOR N=B TO C STEP 10
200       U=(Y-N)*D/(Y-B)
210       MOVE N,(Z+U); DRAW N,(Z-U)
220    NEXT N
230 NEXT B
240 END
```

```
  5REM LISSAJOUS FIGUREN NAAR HCC 48
 10P.$12
 20IN."AMPLITUDE HORIZONTAAL"A
 30IN."AMPLITUDE VERTICAAL  "B
 40FIN."FREQUENTIE VERHOUDING"%V
 50REM INVOER IN GRADEN BEREKENING IN RADIALEN
 60FIN."BEGIN FASE VERHOUDING"%F;%F=%F*PI/180
 70REM HOR.TRILLING ELKE D GRADEN.
 80REM VER.TRILLING ELKE E=D*V GRADEN.
 90%D=PI/90;%E=%D*%V
100CLEAR4
120REM (128,96) IS HET MIDDEN VAN HET SCHERM.
130REM HOR.:U=SIN((N-1)X),Q=SIN(NX),W=COSX
140REM VER.:P=SIN((N-1)X),Q=SIN(NX),W=COSX
150REM VER.:M=COS((N-1)X),N=COSX
160%U=0;%T=SIN(%D);%P=0;%Q=SIN(%E)
170%Z=COS(%D);%W=COS(%E)
180%M=1;%N=%W
190%K=SIN(%F);%L=COS(%F)
200REM LIJN 220 - 310 DE LUS
210REM (128,96) IS HET MIDDEN VAN HET SCHERM
220%S=2*%T*%Z-%U
230%X=A*%S+128
240%U=%T;%T=%S
250%R=2*%Q*%W-%P
260%C=2*%N*%W-%M
270%H=%K*%C+%L*%R
280%Y=B*%H+96
290%M=%N;%N=%C;%P=%Q;%Q=%R
300PLOT13,%X,%Y
310G.220
```

```
100REM LISSAJOUS
110CRMOD
120DIMA1
130%A=0;X=128;Y=96
140IN."PUNTEN OF STREPEN (S/P)"$A
150IF?A=#50;K=9
160IF?A=#53;K=2
170P.$12;?#E1=0;P."LISSAJOUS"
180F.I=0TO400
190PLOT4,X,Y
200%R=100*SIN(2*%A)
210%S=84*COS(3*%A)
220PLOTK,%R,%S
230%A=%A+1
240N.;E.
```

```
  0REM FVAR
  1IN."ROUTINE OP: "Q
  2DIMLL0;LL0=Q
  3P.$21
  4F.N=0TO1;P=Q;[
 10 LDA@0;STA#2887
 15 LDA@#26;STA#28A2
 20 LDA@LL0/256;STA6
 30 LDA@LL0%256;STA5
 40 JSR#C2F2;JMP#C558
 50:LL0
 60];N.;@=0
100$P="F.I=0TO26;P.""%""$I+64""= "";FP.%@@I';N.;E.";P=P+L.P
110P.$6"ROUTINE OP: #"&Q"-#"&P';E.
```

```
10 PROGRAM TEST GRMODE MET EDIT
20 DIMBB4
30 CLS
40 IN."WAAR ASSEMBLER"A
50 F.I=0TO1
60 P=A
70[
80:BB0
90 LDY@3;JSR#ACDE;JSR#AC4B;LDA@#1E;JSR#FFF4
100:BB4
110 RTS
120:BB1
130 JSR#FFE3;CMP@#C;BEQBB4;PHA;LDA#93;CMP@#14;BEQBB3
140:BB2
150 PLA ;JSR#FFE9;JMPBB1
160:BB3
170 PLA;CMP@#D;BEQBB1;CMP@#A;BEQBB1;PHA;LDA@#E0;CMP@#1F;BNEBB2
180 PLAPHA;CMP#B;BEQBB2;CMP@#8;BEQBB2;CMP@#7F;BEQBB2;PLA;JMPBB1
190]
200N.
210E.
```

```
 10 PROGRAM TOWER
 20
 30 DIMHH(3)
 40
 50 PROC HANOI(P,Q,N),R
 60
 70 XIF N)1 THEN
 80      R=6-P-Q
 90      HANOI(P,R,N-1)
100      VERPL(P,Q,N)
110      HANOI(R,Q,N-1)
120 ELSE VERPL(P,Q,1)
130 PEND
140
150
160 PROC TOREN(P,Q,N)
170
180 HH1=0;HH2=0;HH3=0
190 CLEAR4
200 FORI=N TO 1 STEP-1
205  HHP=HHP+1
206  INVERT(P,I)
207 NEXT I
210 HANOI(P,Q,N)
220 PEND
230
240
250 PROC VERPL(P,Q,N)
260 INVERT(P,N);HHP=HHP-1
270 HHQ=HHQ+1;INVERT(Q,N)
280 PEND
290
300
310 PROC INVERT(X,B)
320
330 MOVE(X*85-42),(HHX*4)
340 PLOT0,-B,0;PLOT2,(2*B),0
350 PLOT0,0,1;PLOT2,(-2*B),0
360 PLOT0,0,1;PLOT2,(2*B),0
370 PEND
380
390
400 P.$12''"++++   TOREN VAN HANOI    ++++"
410 P.'''"VERPLAATS EEN AANTAL SCHIJVEN ZO
420 P."DAT ER NOOIT EEN GROTERE SCHIJF"'
430 P."OP EEN KLEINERE LIGT."''
440 INPUT "VAN WELKE PAAL "P
441 IN."NAAR WELKE PAAL "Q
442 IN."AANTAL SCHIJVEN "N
450 P=P&3;Q=Q&3
460 TOREN(P,Q,N)
470 INKEY A;GOTO 400
500
510
```

JAN WIJNEN

```
 10 PROGRAM TURTLE GRAFICS
 20
 30 PROC VOORUIT(N)
 40 PLOT(P&1),%(%X*N),%(%Y*N)
 50 PEND
 60
 70 PROC ACHTERUIT(N)
 80 VOORUIT(-N)
 90 PEND
100
110 PROC HOEK(A),%H
120
130 A=-A
140 %H=%X*COSRADA-%Y*SINRADA
150 %Y=%X*SINRADA+%Y*COSRADA
160 %X=%H
170 PEND
180
190
200 PROC VEEGUIT
210 CLEAR4
220 PEND
230
240 PROC NAARBEGIN
250 MOVE128,96
260 %X=0;%Y=1;P=1
270 PEND
280
290 PROC PENOP
300 P=0
310 PEND
320
330 PROC PENNEER
340 P=1
350 PEND
360
370
380
390 VEEGUIT;NAARBEGIN
400 VOORUIT(50);HOEK(90)
410 VOORUIT(50);HOEK(90)
420 VOORUIT(50);HOEK(90)
430 VOORUIT(50);HOEK(90)
440 HOEK(10)
450 VOORUIT(5)
460 GOTO 400
470
480
490
500 TURTLE GRAFIC'S
510

520 Er zijn een aantal manieren om plaatjes op een computer in
530 elkaar te zetten. De atom gebruikt commando's als plot,draw
540 en move. Dit kan natuurlijk ook anders.
550 Turtle grafics maken het mogelijk plaatjes te maken met
560 behulp van een 'schildpad'.
570 Je kunt deze schildpad over het scherm laten wandelen.
580 De commando's voor die schildpad zijn vrij uitgebreid.
590 Ik heb daar een aantal uit genomen om een idee te geven
600 wat turtle grafics nu beteken.
610 Dat zijn :
620     veeguit    : maak het scherm schoon (clear4)
630     naarbegin  : ga naar begin positie
640     penop      : haal pen van papier
650     penneer    : zet pen op papier
660     vooruit(8): ga 8 punten vooruit
670     hoek(45)   : maak een hoek van 45 graden
680
690 De commando's zijn erg eenvoudig en spreken voor zichzelf.
700 Dit is ook de reden dat ze gebruikt worden in een taal als
710 LOGO, die speciaal voor kinderen gemaakt is.
720
730 In het hoofdprogramma mogen de variabelen p,%x en %y niet
740 gebruikt worden.
750
760
770                          JAN WIJNEN
```

```
 10 PROGRAM HERSENBREKER
 20
 30 DIM T15,R15
 40
 50 PROC DRUKAF,I,J
 60 J=0
 70 FORI=0 TO LENR-1
 80  XIF R?I<65 PRINT $T?J;J=J+1
 90  ELSE PRINT $R?I
100 NEXT;PRINT " "
110 IF COUNT+LENR>30 PRINT '
120 PEND
130
140 PROC BREKER(C,F),I
150 I=1
160 IF C=1 IF F=M DRUKAF
170 IF F=M OR ?#B001&128=0 GOTO 210
180 IF C%I=0;T?F=64+I;F?#8000=I;BREKER(C/I,F+1)
190 I=I+1
200 IF I<27 GOTO 180
210 PEND
220
230 PROC HERSEN(C,R),I,M
240 M=LENR
250 FOR I=0 TO LEN R-1
260  IF I?R>64 C=C/(I?R-64);M=M-1
270 NEXT
280 M!#8000=#20202020
290 BREKER(C,0)
300 PEND
310
320 PRINT $12"-=-=-=-=- HERSENBREKER -=-=-=-=-"'
330 INPUT "GETAL "C,"WOORD "$R
340 HERSEN(C,R)
350 IF COUNT>0 PRINT '
360 GOTO 330
370
380 ++++++++  HERSENBREKER ++++++++
390
400 DIT PROGRAMMA GENEREERT ALLE LETTER COMBINATIES
410 VAN EEN HERSENBREKER.
420 (ZIE DENKSPORT PUZZELS.)
430
440 JE GEEFT EERST HET GETAL EN DAN HET WOORD, ZOVER
450 JE DE LETTERS AL HEBT.
460 VOORBEELD :
470 GETAL ?4560 (RETURN)
480 WOORD ?...L (RETURN)
490 MET ALS UITVOER OA HET WOORD STAL, DAT GEZOCHT WERD.
500 DUURT HET KRAKEN VAN HET GETAL TE LANG, DRUK DAN SHIFT.
510
520 VEEL HERSENBREKERPLEZIER GEWENST !!
530
540 JAN WIJNEN   BRABANT OOST
```

```
 10 PROGRAM 3-D
 20 FDIM %@@6
 30 PROC POSITIE(%X,%Y,%Z)
 40  %@@0=%X;%@@1=%Y;%@@2=%Z
 50  %@@3=%X*%X+%Y*%Y;%@@4=SQR%@@3
 60  %@@5=%@@3+%Z*%Z;%@@6=SQR%@@5
 70 PEND
 80
 90 PROC PROJECTEER(%X,%Y,%Z:P,Q),%O
100  %O=%@@5-%X*%@@0-%Y*%@@1-%Z*%@@2;%O=%O*%@@4
110  P=%(400*(%Y*%@@0-%X*%@@1)*%@@6/%O)+128
120  Q=%(500*(%Z*%@@3-%@@2*(%X*%@@0+%Y*%@@1))/%O)+96
130 PEND
140
150 P.$12
160 INPUT"POSITIE :"'"X "X,"Y "Y,"Z "Z
170 CLEAR4
180 POSITIE(X,Y,Z)
190 RESTORE
200 READ N
210 WHILE N()-99
220  READ M,O,P
230  PROJECTEER(N,M,O,X,Y)
240  PLOT P,X,Y
250  READ N
260 WEND
270 INKEY A
280 G.150
290 DATA -10,-6,-6,4
300 DATA 10,-6,-6,5
310 DATA 0,0,12,5
320 DATA -10,-6,-6,5
330 DATA 0,6,-6,5
340 DATA 0,0,12,5
350 DATA 0,6,-6,4
360 DATA 10,-6,-6,5
370 DATA -99
380
390 3-D routine
400 Het maken van 3 dimensionale plaatjes op een computer is
410 een nogal gecompliceerde bezigheid.
420 Dat komt vooral door de grote hoeveelheid gegevens die je
430 bij moet houden.
440 Deze routine neemt dat werk van je over.
450 Je kiest eerst een positie tov de oorsprong(0,0,0)
460 bv. positie(10,100,-30)
470 Daarna kun je de projectie van een 3-d punt op je 2-d tv
480 opvragen met projecteer.
490 bv. projecteer(1,-1,3,P,Q)
500 Het projectiepunt staat dan in p,q.
510 In het voorbeeld is dat P=124 EN Q=108.
520 De routine gebruikt %@@0 tot %@@6.
530 Het kan zijn dat het programma je bekend voorkomt, het is
540 namelijk gebaseert op het 3-d programma uit het manual,
550 pagina 166.
560 Jan Wijnen     Brabant Oost
```

```
 10 PROGRAM DRIEHOEK
 20
 30 PROC TRIANGEL(X,Y,A),B
 40 RESTORE 170
 50 B=A/2
 60 MOVEX,Y
 70 FOR I=0 TO 17
 80 READ P,Q
 90 CASE I OF
100(10) PLOT0,P,Q
110(14) PLOT0,P,Q
120(I) PLOT3,P,Q
130 CEND
140 NEXT I
150 PEND
160
170 DATA 0,8*A,A,B,8*A,-4*A,0,-A,-8*A,-4*A
180 DATA -A,B,A,-B,0,7*A,A,-B,0,-6*A
190 DATA 7*A,4*A+B,-7*A,-3*A-B,0,A,6*A,3*A,-8*A,3*A
200 DATA 7*A,-3*A-B,-A,-B,-6*A,3*A
210
220 CLEAR4
230 !#8000=-1;I=#8000;J=#8003;DOCOPYI,J-1,J;J=J+J-I;U.J)=#9800
240 TRIANGEL(50,20,20)
250 END
260
270
280
290
300 De onmogelijke driehoek.
310
320 Escher is een graficus diebij de meeste mensen bekend is om
330 zijn 'onmogelijke' tekeningen. Hij tekende bijvoorbeeld een
340 trap die altijd naar boven loopt en toch op het zelfde punt
350 uitkomt. Een huis waarin muren plafonds, plafonds vloeren
360 en vloeren muren schijnen te zijn.
370 Deze tekeningen zijn gemaakt door de perspectief wetten op
380 een verdraaide manier te gebruiken.
390 Zo ook deze driehoek. Hierbij lijkt de ene zijde naar
400 achter te gaan en de andere naar voren terwijl ze toch aan
410 elkaar aansluiten.
420 Rara, hoe kan dat !!
430
440
450                    JAN WIJNEN
```

```
10 REM *** ONMOGELIJK FIGUUR ***
20 REM EDWIN HOBBEL
30 CLEAR4
40 MOVE96,26;DRAW160,40
50 MOVE160,40;DRAW180,140
60 MOVE96,26;DRAW116,126
70 MOVE116,126;DRAW122,116
80 MOVE116,126;DRAW163,136
90 MOVE96,26;DRAW90,35
100 MOVE90,35;DRAW110,135
110 MOVE110,135;DRAW174,149
120 MOVE174,149;DRAW180,140
130 MOVE174,146;DRAW154,46
140 MOVE154,46;DRAW107,36
150 MOVE154,46;DRAW148,55
160 MOVE107,36;DRAW122,166
170 MOVE163,136;DRAW148,55
180 MOVE148,55;DRAW110,47
190 MOVE122,116;DRAW160,124
200 LINK#FFE3
210 I=#8000;N=#9800
220 !I=!I:-1;I=I+4;!N=!N:-1;N=N-4
230 IF N<#8000 IF I>#9800 THEN GOTO 200
240 IF N=#8C00 THEN GOTO 260
250 GOTO 220
260 !#8C00=!#8C00:-1;LINK#FFE3
270 !#8C00=!#8C00:-1;GOTO 220
```

```
10 REM *** 3D-PLOT ***
20 REM EDWIN HOBBEL
30 A=38;B=6;C=218;D=186;X=0
40 CLEAR4;GOTO 60
50 X=X+8
60 MOVE(A+X),B;DRAWC,(B+X)
70 MOVEC,(B+X);DRAW(C-X),D
80 MOVE(C-X),D;DRAWA,(D-X)
90 MOVEA,(D-X);DRAW(A+X),B
100 IF X>=175 THEN GOTO 120
110 GOTO 50
120 FOR I=#8000 TO #9800 STEP 4
130 !I=!I:-1;NEXT
140 LINK#FFE3;GOTO 120
```

Dit is niet de snelste, maar wel de simpelste manier om het grafische scherm wit te maken:

```
GR.;COPY #8000,#9800,#8001
```

Voor uitleg zie de bijdrage van Bram Poot, A.N. 6,1983,blz. 98

Frank de Groot

```
10P.$12$15"OMREKENING VAN QTH-LOCATOR NAAR"'
20P."LENGTE- EN BREEDTEGRADEN"'''
30REM**OORSPRONG : ONBEKEND
40REM**VOOR ACORN ATOM BEWERKT DOOR :
50REM**HANS SCHOON, POSTBUS 7020, 3130 JA VLAARDINGEN
60REM**TEL. 010-742904
70P."DIT PROGRAMMA ZET DE (NIEUWE)"'
80P."QTH-LOCATOR OM IN LENGTE- EN "'
90P."BREEDTEGRADEN, GEBASEERD OP HET"'
100P."MIDDELPUNT VAN HET LOCATORVAK."'''
110DIMQ5
120IN."GEEF (RETURN)"$Q;P.$12
130IN."ww QTH-LOCATOR "$Q
140IF$Q=""P.$12"EINDE"'&TOP';E.
150A=Q?0-65;B=Q?1-65;C=Q?2-48;D=Q?3-48;E=Q?4;F=Q?5
160IFA<0ORB<0ORC=0ORD<0ORE<65ORF<65ORLENQ<5P.'"ONJUIST"'';G.130
170%G=(E-64.5)/12
180%H=(F-64.5)/24
190%I=-180+A*20+C*2+%G
200%J=-90+B*10+D+%H
210FP.'"LENGTEGRAAD : "
220%X=ABS(%I);GOS.a
230FIF%I<0P." W.L."'
240FIF%I>0P." O.L."'
250FP."BREEDTEGRAAD: "
260%X=ABS(%J);GOS.a
270FIF%J<0P." Z.B."
280FIF%J>0P." N.B."
290P.''';G.130
300a@=4;W=4;%@=100*W+@;W=%(10↑W+1)
310Y=%(W*ABS%X+.5)-ABS%X*W;IFY=W;P.%X+SGN%X".";Y=0;G.b
320FIF%X<0;FIF%X>-1;DOP." ";U.C.>=@-2;P."-";@=0
330P.%X"."
340b@=G;DO IFY<W/10;P.0
350W=W/10;U.W=1;IFY>0;P.Y
360W=%@/100;@=%@%100;R.
```

VOORBEELDEN :

LOCATOR JO21EW WORDT : 4.375 O.L./51.9375 N.B.

LOCATOR JO86DQ WORDT : 16.2917 O.L./56.6875 N.B.